マイコンBASICマガジン 1月号別冊付録

プログラミング派パソコン・ユーザー必携

高速・高機能BASIC言語リファレンス・マニュアル

(N88 BASIC/True BASIC/Quick BASIC)

```
50 GOSUB 400'違うときの処理
60 GOTO 20
70 GOSUB 500'ゲーム終了
80 IF D=1 THEN GOTO 10
90 END
100 PRINT "数当てげぃむ"
110 C=1
120 A=INT(RND(1)*100)+1
130 RETURN
200 INPUT B
210 RETURN
300 IF A=B THEN PRINT "正解! ";C;"回でした"
310 IF A>B THEN PRINT "もっと大きい"
320 IF A<B THEN PRINT "もっと小さい"
330 RETURN
400 C=C+1
410 PRINT "もう一度チャレンジ"
420 RETURN
500 PRINT "再プレー?"
510 I$=INKEY$
520 IF I$="Y"
```

```
35 '入力と結果表示
40 INPUT B
50 IF A=B THEN PRINT
60 IF A>B THEN PRINT
70 IF A<B THEN PRINT
75 '正解しているなら終
80 IF A=B THEN GOTO 1
85 '違うとき
90 C=C+1
100 PRINT "もう一度チ
110 GOTO 40
115 'ゲーム終了
120 PRINT "再プレー
130 I$=INKEY$
140 IF I$="Y" OR I$
150 IF I$="N" OR I$
160 GOTO 130
```

あなたに優しいBASIC

●Written By PWM

パソコンでプログラムを作るとき，あなたはどういう言語を使っていますか？

最近はC言語がブームとなり，さまざまな高性能Cコンパイラが登場しています。これらのコンパイラは統合環境といったエディタとコンパイラが合体した構成を取っていてたいへんに扱いやすいものです。

しかし，これらの工夫されたCコンパイラでも，取っつきやすさという点ではまだまだBASICには及びません。BASICのほとんどはパソコン本体に標準装備されていますし，インタプリタ形式のものがほとんどですから親しみやすさに関してはこれ以上のものはないでしょう。

親しみやすいぶん，プログラムの実行速度が遅く，市販ソフトのような大規模なプログラムは作りにくいなどの問題点がありますが，手軽にプログラミングをしてみる場合はBASICが一番ではないでしょうか。

取っつきやすいBASIC

BASICの親しみやすさを見てみるために，同じ内容を実行するBASICのプログラムとCのプログラムを**リスト1**，**リスト2**に用意しました。

このプログラムはどちらもキーボードから入力される二つの数値によって足し算を行ない，その和を表示する，という簡単なものです。

リスト1と**リスト2**を比べてみると，Cのプログラムである**リスト2**のほうには難しそうな設定，意味が良くわからない記号などが含まれていることがわかります。

これに対して，BASICのプログラムである**リスト1**は，命令が英語であることを除けばたいへんわかりやすいものになっています。

プログラミングというものがどういうものかまだわからない人が，いきなり**リスト2**のような複雑そうなプログラムを書けるでしょうか？

リスト2のプログラムの実行内容は非常に簡単ですが，内部動作についてはポインタなどのCPUやメモリ構造の考えかたをすべて把握していないと完全に理解するのは不可能です。

BASICのプログラムは他の言語と同様，命令は英語です。しかし，その単語の意味さえわかっていれば，その命令の内容は理解できてしまいます。さらに，複雑な指定などがあまりないので，命令の動作を完全に理解するのも簡単です。これが，BASICの取っつきやすさを生みだしている大きな理由です。

C言語やPascalなどは一つの命令を実行するために，指定したり用意しなくてはならない項目が多く，その動作内容もかなり複雑です。プログラムを組みはじめる前に，プログラムを組むためにやらなくてはならないことが多いのです。

これでは，初心者はプログラムができあがる前に設定などでつまづいてしまいます。ところが，BASICならば実行したい命令を並べていけば，簡単にプログラムができてしまうのです。この違いがBASICとC言語との大きな違いであり，BASICの特徴です。

インタプリタとコンパイラ

ベーマガの読者のみなさんの大部分は，「BASICインタプリタ」とか「Cコンパイラ」という言葉を見聞きしたことがあると思います。BASICやC言語というのはコンピュータのプログラミング言語の種類ですが，インタプリタやコンパイラというのはいったいどういうものなのでしょうか？

BASICというとインタプリタ，C言語といえばコンパイラというように考えられている風潮がありますが，BASICは必ずインタプリタで，Cは全部コンパイラかというとそれは間違いです。

BASICコンパイラは各種存在しますし，Cインタプリタもあります。言語の種類とインタプリタかコンパイラかというのはまったく独立しているのです。

全体の傾向としては，BASICはインタプリタが多く，Cはコンパイラがほとんどですので，BASIC＝インタプリタ，C＝コンパイラという言葉が良くでるわけです。

BASICではよく採用されているインタプリタ方式では，インタプリタと呼ばれるプログラムが，与えられたプログラムを逐次解析して対応する処理を実行していきます。

たとえば，**リスト3**のようなプログラムの場合，インタプリタは**第1図**に示すような手順でプログラムを実行し

《LIST 1》BASICによる足し算プログラム

```
10 INPUT A          ' 変数Aにキーボードから値を入力する
20 INPUT B          ' 変数Bにキーボードから値を入力する
30 C=A+B            ' 変数CにAとBの和を代入する
40 PRINT C          ' Cの値を画面に表示する
50 END              ' 終わり
```

《LIST 2》Cによる足し算プログラム

```
#include     <stdio.h>      /* ヘッダファイルを読み込む */

int     a;                  /* 各変数を宣言する */
int     b;
int     c;

main()                      /* メインプログラム */
{
        scanf("%d",&a);     /* aに標準入力から値を読み込む */
        scanf("%d",&b);     /* bに標準入力から値を読み込む */
        c=a+b;              /* cにaとbの和を代入する */
        printf("%d¥n",c);   /* 標準出力にcの値を表示し、改行する */
```

《LIST 3》インタプリタとコンパイラの動作内容

```
10 S=0
20 INPUT A
30 S=S+A
40 GOTO 20
```

ていきます。

いっぽう，コンパイラ方式はどうかと言うと，**第2図**に示すような手順でプログラムを実行します。

コンパイラは，与えられたプログラムに対応する実行オブジェクトを作りだすだけで，そのプログラム自体を実行することはありません。できあがったオブジェクトをOS上などで実行して，はじめてプログラムが実行されたことになります。

インタプリタはプログラムに書かれている命令に対応する動作をその都度実行していきます。ですから，ループする部分などがある場合は同じところを何度も解析して実行するという無駄なことをします。このため，速度はかなり低下してしまいます。

コンパイラの場合はあらかじめプログラム全体を解析してしまうため，実行しているときには解析する必要がありません。このため，コンパイラで作った実行オブジェクトは高速です。

そのかわり，インタプリタではプログラムを即実行できますが，コンパイラの場合はプログラム実行前にコンパイルという作業を行なっておく必要があります。

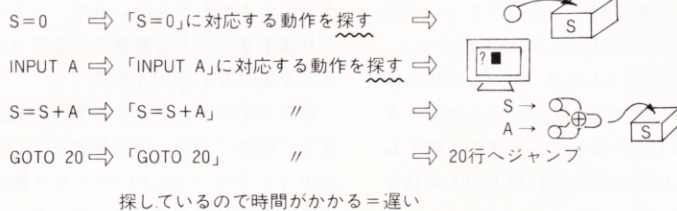
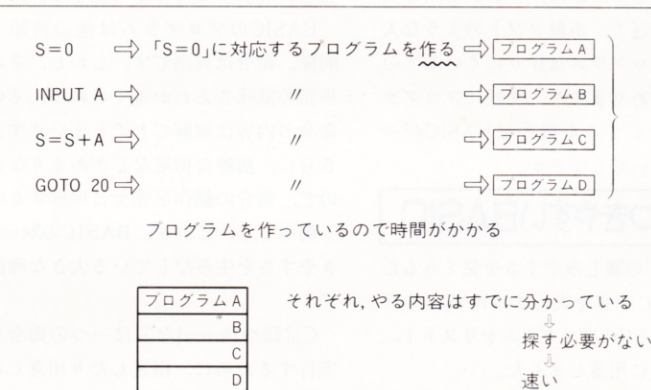
簡単にプログラムを操作できるインタプリタがBASICに多いのは，BASIC自体が持つ親しみやすさを損なわないようにしているためです。

一般には速度の点で劣っているインタプリタはほとんど使われませんが，パソコン・レベルのBASICの世界ではこのような理由からインタプリタが主流となっています。

◀ROM−BASICはハードディスクやフロッピーディスクから起動しないようにしていると起動する。FILESなどのディスク操作命令が使えないため，現在ではほとんど用いられていない

《第1図》インタプリタの動作

《第2図》コンパイラの動行

構造化プログラミング

このように親しみやすさに重点をおいたBASICですが，親しみやすさを優先したために大きな欠点を持っています。

それが，構造化プログラミングがしにくいということです。

小さなプログラムを作る場合は，プログラム全体を簡単に見ることができ

```
10 C=1
20 PRINT "数当てげぃむ"
30 A=INT(RND(1)*100)+1
40 INPUT B
50 IF A=B THEN PRINT "正解!";C;"回でした":GOTO 110
60 IF A>B THEN PRINT "もっと大きい":GOTO 80
70 PRINT "もっと小さい"
80 C=C+1
90 PRINT "もう一度チャレンジ"
100 GOTO 40
110 PRINT "再プレー?"
120 I$=INKEY$
130 IF I$="Y" OR I$="y" THEN GOTO 10
140 IF I$="N" OR I$="n" THEN END
150 GOTO 120
```

《LIST4》わかりにくいプログラム

```
5 '初期化
10 PRINT "数当てげぃむ"
20 C=1
30 A=INT(RND(1)*100)+1
35 '入力と結果表示
40 INPUT B
50 IF A=B THEN PRINT "正解! ";C;"回でした"
60 IF A>B THEN PRINT "もっと大きい"
70 IF A<B THEN PRINT "もっと小さい"
75 '正解しているなら終了
80 IF A=B THEN GOTO 120
85 '違うとき
90 C=C+1
100 PRINT "もう一度チャレンジ"
110 GOTO 40
115 'ゲーム終了
120 PRINT "再プレー?"
130 I$=INKEY$
140 IF I$="Y" OR I$="y" THEN GOTO 10
150 IF I$="N" OR I$="n" THEN END
160 GOTO 130
```

《LIST5》ブロックごとに分けたプログラム

ます。ですから，プログラムの構造もすぐにわかります。

ところが，プログラムが大規模になるにしたがって，プログラム全体を見通すことが難しくなり，構造をつかみにくくなります。

一般に一つの処理が1画面中に表示しきれなくなると，急に見通しが悪くなってプログラムが作りにくくなります。

このようなときは，プログラムを機能単位で分割して，見通しをよくするという手段が用いられます。

たとえば，プログラム全体を一つの文章とすると，だらだら続く長い文章は読みにくいものですが，章や段落などで内容をブロック化することによって格段に読みやすくなります。

このブロックに名前を付けて目次にすれば，全体の見通しも良くなります。プログラムでも同じことがいえます。

だらだらと，さまざまな処理が入り乱れている**リスト4**のようなプログラムは非常に読みにくいものです。

しかし，これを**リスト5**のように処理内容ごとにブロック化すれば見やすくなります。

リスト6のように処理内容ごとにサブルーチン化してしまえば，さらに見通しが良くなります。

このように，処理をブロック化して見やすいプログラムを作ることを構造化プログラミングといいます。

BASICにはこのような構造化プログラミングをするために必要な機能が完全には備わっていません。ですから，完全な構造化プログラミングはできないのです。

でも，構造化を考えつつプログラミングするのと，何も考えずにプログラミングするのとでは，できあがりのプログラムの読みやすさが違います。また，構造化を考えていれば，大きなプログラムになるほどプログラミングがしやすくなります。

LISTコマンドで1画面に納まるようなプログラムなら，構造化は考えなくてもいいでしょう。1画面に納まる程度なら見通しもききますし，どこで何をしているかということを把握するのも簡単です。

しかし，それを超える大きさのプログラムを作る場合は，構造化を考えて見通しよくプログラミングするほうが簡単確実に質の高いプログラムを作れます。

◀DISK-BASIC(日本語BASIC)はPC-9801用の主流と言えるBASIC。ROM-BASICにディスク関係の命令などを拡張し、ファイル操作や日本語処理などができる

《第3図》N88-BACIC

《LIST6》サブルーチン化したプログラム

```
10 GOSUB 100'初期化
20 GOSUB 200'入力
30 GOSUB 300'結果表示
40 IF A=B THEN 70
50 GOSUB 400'違うときの処理
60 GOTO 20
70 GOSUB 500'ゲーム終了
80 IF D=1 THEN GOTO 10
90 END
100 PRINT "数当てげぃむ"
110 C=1
120 A=INT(RND(1)*100)+1
130 RETURN
200 INPUT B
210 RETURN
300 IF A=B THEN PRINT "正解！ ";C;"回でした"
310 IF A>B THEN PRINT "もっと大きい"
320 IF A<B THEN PRINT "もっと小さい"
330 RETURN
400 C=C+1
410 PRINT "もう一度チャレンジ"
420 RETURN
500 PRINT "再プレー？"
510 I$=INKEY$
520 IF I$="Y" OR I$="y" THEN D=1:RETURN
530 IF I$="N" OR I$="n" THEN D=0:RETURN
540 GOTO 510
```

主要BASICいろいろ

●N88-BASIC

PC-9801シリーズに標準装備されているBASICです。

このBASICは名前のとおり，当初はPC-8801用のBASICとして登場しました（ちょうど10年前）。その後，PC-9801にはPC-8801との互換性を保つという意味で，N88-BASIC(86)として搭載されました。

このころはまだメインメモリが64kBから256kB程度，記憶装置は2Dのフロッピーディスクやカセット・テープでした。

ほとんどのユーザーはプログラムを作るとカセット・テープにできたプログラムを何分もかけてセーブしていました。このころは，まだフロッピーディスク・ドライブは高価で，フロッピーディスクが1枚で1,000円程度（10

枚で１万円！）もしていたのです。

現在，ROM-BASICと呼ばれているBASICがあります。

PC-9801シリーズのマシンに，ハードディスクを接続せず，フロッピーディスクもセットしないでしばらく放っておくと，BASICが起動します。これがROM-BASICなのですが，これが当時は主力BASICとして利用されていました。

現在はほんの一部のソフトの互換性確保のために残っているだけで，フロッピーディスクなどへのアクセスの機能がないため実用にはなりません（エプソンの互換機には装備されていない）。

フロッピーディスク・ドライブが一般的になり，ほとんどのユーザーがフロッピーディスクを利用するようになってからは，ROM-BASICにディスク関係の命令を拡張したDISK-BASICが主流となりました。

このBASICはディスク管理や日本語処理が可能であり，十分実用になりました。日本語が利用できる点から日本語BASICと呼ぶ場合もあります。

現在PC-9801シリーズに標準装備されているBASICがこれです。このBASICは国内では標準的ともいえるもので，対応ソフトも数多く制作されました。現在でもさまざまなところで利用されています。

ディスク・ドライブが普及してMS-DOSが移植されてからは，MS-DOSを利用するユーザーが急激に増えてきました。このMS-DOSに対応するN88-BASICとして，MS-DOS版N88-BASIC（86）が登場しました。

このBASICはファイル管理などはMS-DOSに対応させてあり，それ以外は本来のDISK-BASICとコンパチになっています。

さらに，対応するBASICコンパイラも登場し，BASIC上でプログラムを開発した後，コンパイラで実行速度をアップすることもできるようになりました。

このように，N88-BASICには，大きくわけて三つのBASICがあります。本体に内蔵されているROM-BASIC，標準装備のDISK-BASIC，MS-DOS対応のMS-DOS版BASICの三つです。

◀DISK-BASICをMS-DOSに対応させたBASICがMS-DOS版N88-BASIC。ファイル操作などはMS-DOSを通して行なわれるため、ファイル関係は互換性が一部取れていない。MS-DOSの上で実行するため、MS-DOSのぶんだけフリーエリアが小さくなる。コンパイラが用意されていて、高速化ができる

◀統合環境をサポートし、コンパイラなどが利用できるようになっているQuick BASIC。BASICとなっているが、プログラミング・スタイルなどはC言語によく似ている

Quick BASICには強力なオンライン・ヘルプが登載されていて、マニュアルを見なくても操作ができる

●Quick BASIC

MS-DOSが一般的になってからは，MS-DOS上のCコンパイラやPascalコンパイラ，Fortranコンパイラなどが普及し，C言語がブームとなってTurbo CやQuick Cといった統合環境を備えた使いやすいCコンパイラが登場しました。

Quick BASICはこのような中で，BASICの欠点を改善した新しいBASICとして登場した新世代BASICです。

■Quick BASIC問い合わせ先：マイクロソフト（株）〒105 東京都港区芝1-11-11 住友不動産芝ビル ☎03-5484-0700

BASICは非構造化言語であり，大規模なプログラムを作るのが難しくなっています。

Quick BASICではこの欠点を完全になくし，構造化されたBASICプログラムを作りだせます。また，C言語で流行の統合環境をサポートし，インタプリタ，エディタ，コンパイラが統合環境内部で利用できます。強力なオンライン・ヘルプによって，わからない点などはマニュアルを見なくても調べられます。

Quick BASICはBASICを構造化したというよりも，C言語をBASICっぽくしたというほうが適切でしょう。

まず，BASICプログラムでは常識の行番号がありません。当然，GOTO文やGOSUB文はありません。さらに，C言語に備わっているループ構造が名前こそ違いますがほぼそのまま用意されています。

いっぽう，従来のBASICでのループ構造を作るノウハウの大部分は使えなくなっていて，BASICに慣れているからといってQuick BASICがすぐに使えるようになるとは限りません。

C言語の影響を大きく受けているBASICです。

●True BASIC

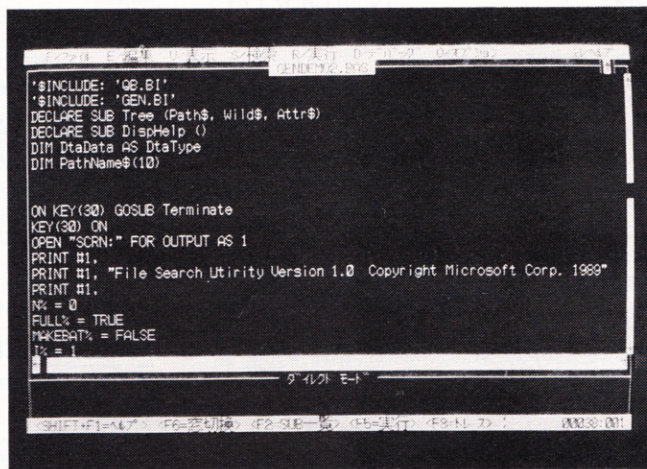
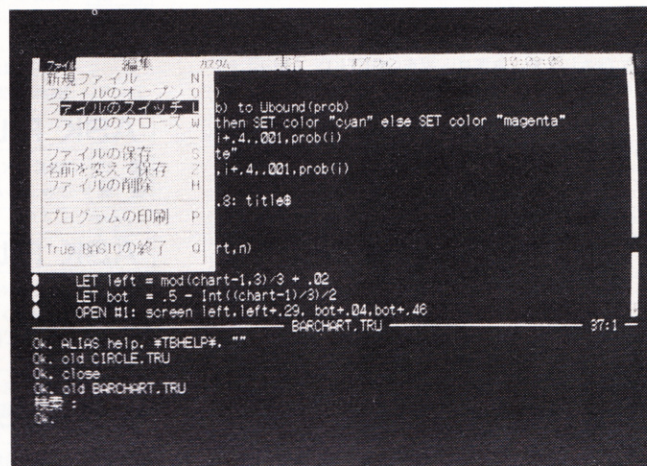
True BASICはBASIC言語の創始者のケメニー・カーツ博士が自ら開発した新世代BASICです。

Quick BASICがC言語の影響を受けることによって構造化しているのとは違い，True BASICはBASICをベースにさまざまな言語にみられる構造化手法を取り入れて構造化されています。

同時に，開発者がBASICの創始者であるため，BASICの特徴がかなり残っています。たとえば，行番号やGOTO文，GOSUB文などが用意されていますし，DATA文やREAD文，ON GOTO・GOSUB文などのBASIC独特の命令もあります。ですから，今までのBASICと同じ感覚で利用できます。

このようなBASICの特徴を残した上で，外部関数や構造化ループ命令，構造化条件判断命令などの構造化プログラミングに必要な各種命令を取り入れています。これらの命令を利用すれば，完全に構造化されたプログラムを書けます。

True BASICもQuick BASIC同様，流行の統合環境になっていて，インタプリタ・エディタ・コンパイラが用意されています。実行画面とプログラム編集画面が独立していて，プログラムを見ながら実行結果を見ることができます。

従来のBASICの雰囲気を保ったまま，最新の構造化言語として必要な命令を拡張し，流行のスタイルにしたというおもしろいBASICです。

◀Quick BASICのプログラムは通常のBASICとは大幅に異なっている。行番号などは使わない。構造としてはC言語のものをBASICに当てはめたような構造になっている

◀従来のBASICで用いられているプログラミングスタイルでのプログラミングも，構造化したプログラミングも，どちらも可能なTrue BASIC。統合環境をサポートしている

```
! 円のピクチャを描きます。
!
PICTURE Circle(sides)

    FOR i = 0 to sides
       LET u = (i*2*Pi)/sides       ! Find next angle
       PLOT Cos(u), Sin(u);         ! Plot next segment
    NEXT i
    PLOT                            ! Turn off beam

END PICTURE

SET WINDOW -15, 15, -10, 10        ! Set the window
ASK MAX COLOR cmax                 ! Maximum colors

FOR j = 1 to 30                    ! Draw a bunch of circles
                         CIRCLE.TRU
Copyright (c) 1991 by True BASIC, Inc.  All rights reserved.
Ok. ALIAS do, ¥TBDO¥, ""
Ok. ALIAS library, ¥TBLIBS¥, ""
Ok. ALIAS help, ¥TBHELP¥, ""
Ok. old CIRCLE.TRU
Ok.
```

◀True BASICは構造化したプログラミングが可能だが，行番号方式でもプログラミングできる

■True BASIC問い合わせ先：(株)クレオ 〒108 東京都港区芝浦4-9-25 芝浦スクエアビル ☎03-3456-4650

■本リストは、日本電気、クレオ、マイクロソフト各社の製品マニュアルをもとにまとめたものです。

ABS関数　N88/QB/TB

書式　ABS(〈数値式〉)

機能　〈数値式〉の絶対値を返します。

ACOS関数　TB

書式　ACOS(〈数値式〉)

機能　〈数値式〉のアークコサイン（逆余弦）の値を返します。

AKCNV$ (DISKモード)　N88

書式　AKCNV$(〈文字列〉)

機能　1バイト系の英数カナ文字を，対応する2バイト系全角文字に変換します。

ALIAS　TB

機能　［引数なし］ALIASコマンドの指定内容を表示します。
［引数一つ］ファイルの種類による指定内容を表示します。
［引数二つ以上］いろいろな種類のファイルが入っているディレクトリを指定します。

ANGEL関数　TB

書式　ANGEL（〈数値式1〉，〈数値式2〉）

機能　X軸の正の部分と，原点から点ANGEL（〈数値式1〉，〈数値式2〉）にひいた線分とがつくる角を，反時計回りに計った角度で返します。角度は，現在のOPTION ANGELが，RADIANS（デフォルト）のときはラジアンで，DEGREESのときは度数で表されます。

ANK$関数　TB

書式　ANK$(〈文字列式〉)

機能　〈文字列式〉の中に2バイト系全角文字の英数字，カタカナ，句読点があると，それらを1バイト系半角文字に変換して返します。

ASC　N88/QB

書式　ASC（〈文字列〉）

機能　文中のキャラクター・コードを得ます。

ASIN関数　TB

書式　ASIN〈数値式〉

機能　〈数値式〉のアークサイン（逆正弦）の値を返します。

ASK ACCESS　TB

書式　ASK #（整数式）：ACCESS〈文字列変数式〉

機能　ファイルのアクセス・モードを調べ〈文字列変数式〉に代入します。

ASK BACKGROUND COLOR文　TB

書式　ASK BACK GROUND COLOR〈数値変数〉
ASK BACK〈数値変数〉
ASK BACKGROUND COLOR〈文字列変数式〉
ASK BACK〈文字列変数式〉

機能　〈数値変数〉または〈文字列変数式〉に現在の画面のフォアグラウンド・カラーの番号または名前を代入します。

ASK COLOR文　TB

書式　ASK COLOR〈数値変数〉
ASK COLOR〈文字列変数式〉

機能　〈数値変数〉または〈文字列変数式〉に現在の画面のフォアグラウンド・カラーの番号または名前を代入します。

ASK COLOR MIX文　TB

書式　ASK COLOR MIX文（〈整数式〉）〈数値変数1〉，〈数値変数2〉，〈数値変数3〉

機能　〈整数式〉で指定した番号の色の赤，緑，青の成分を代入します。

ASK CURSOR文　TB

書式　ASK CURSOR〈文字列変数式〉
ASK CURSOR〈数値変数1〉，〈数値変数2〉

機能　カーソルの状況を次のように〈文字列変数式〉に代入します。
ON　カーソルが表示，またはグラフィックが使えない場合
OFF　カーソルが非表示
または，カーソルの現在の行と桁の位置を，〈数値変数1〉，〈数値変数2〉代入します。

ASK DATUM文　TB

書式　ASK #〈整数式〉：DATUM〈文字列変数式〉

機能　ストリーム・ファイルの中の次の項目の型を〈文字列変数式〉に代入します。
NUMERIC　数値
STRING　文字列
NONE　なし
UNKNOWN　不明，またはストリーム・ファイル以外

ASK DIRECTORY文　TB

書式　ASK DIRECTORY〈文字列変数式〉

機能　カレント・ディレクトリの名前を〈文字列変数式〉に代入します。

ASK ERASABLE文　TB

書式　ASK #〈整数式〉：ERASABLE〈文字列変数式〉

機能　ファイルが削除できるかどうかを〈文字列変数式〉に代入します。
YES　ERASE文で削除できる
NO　YES以外

ASK FILESIZE文　TB

書式　ASK #〈整数式〉：FILESIZE〈数値変数〉

機能　ファイルのレコード数またはバイト数を〈数値変数〉に代入します。

ASK FILETYPE文　TB

書式　ASK #〈整数式〉：FILETYPE〈文字列変数式〉

機能　〈整数式〉がファイルを参照しているかどうかを〈文字列変数式〉に代入します。
FILE　ファイルを参照
DEVICE　FILE以外

ASK FREE MEMORY文　TB

書式　ASK FREE MEMORY〈数値変数〉

機能　未使用メモリのうち，利用可能なバイト数を〈数値変数〉に代入します。

ASK LANGUAGE文　TB

書式　ASK LANGUAGE〈文字列変数式〉
ASK LANG〈文字列変数式〉

機能　プログラムの中で出力するメッセージのモードを〈文字列変数式〉に代入します。

ASK MARGIN文　TB

書式　ASK MARGIN〈数値変数〉
ASK #〈整数式〉：MARGIN〈数値変数〉

機能　ウィンドウやファイルのマージンを〈数値変数〉に代入します。

ASK MAX COLOR文　TB

書式　ASK MAX COLOR〈数値変数〉

機能　フォアグラウンド・カラーの最大の色番号を〈数値変数〉に代入します。

ASK MAX CURSOR文　TB

書式　ASK MAX CORSOR〈数値変数1〉，〈数値変数2〉

機能　カレント・ウィンドウの最大行数と最大桁数を，〈数値変数1〉〈数値変数2〉に代入します。

ASK MODE文　TB

書式　ASK MODE〈文字列変数式〉

機能　スクリーン・モードを〈文字列変数式〉に大文字で代入します。

ASK NAME文 TB

書式 ASK NAME〈文字列変数式〉

機能 ASK #〈整数式〉: NAME〈文字列変数式〉
プログラム名またはファイル名を〈文字列変数式〉に代入します。

ASK ORGANIZATION文 TB

書式 ASK #〈整数式〉: ORGANIZATION〈文字列変数式〉
ASK #〈整数式〉: ORG〈文字列変数式〉

機能 ファイルの構造を調べ〈文字列変数式〉に代入します。
TEXT テキスト・ファイル
STREAM ストリーム・フエイル
RANDOM 可変長ファイル
RECORD 固定長ファイル
BYTE バイトファイル
WINDOW ウィンドウ

ASK PIXELS文 TB

書式 ASK PIXELS〈数値変数1〉,〈数値変数2〉

機能 縦方向, 横方向の画素の数を〈数値変数1〉〈数値変数2〉に代入します。

ASK POINTER文 TB

書式 ASK #〈整数式〉: POINTER〈文字列変数式〉

機能 ファイルのポインタの位置を調べ〈文字列変数式〉に代入します。
BEGIN ファイルの最初
END ファイルの終わり, またはファイルが空
MIDDLE ファイルの最初と終わり以外, またはウィンドウ

ASK RECORD文 TB

書式 ASK #〈整数式〉: RECORD〈数値変数〉

機能 ファイルのポインタの現在の位置をレコード単位またはバイト単位で〈数値変数〉に代入します。

ASK RECSIZE文 TB

書式 ASK #〈整数式〉: RECSIZE〈数値変数〉

機能 ファイルのレコード長をバイト単位で〈数値変数〉に代入します。

ASK RECTYPE文 TB

書式 ASK #〈整数式〉: RECTYPE〈文字列変数式〉

機能 ファイルのレコードの型を〈文字列変数式〉に代入します。
DISPLAY テキスト・ファイル, ウィンドウ, 画面
INTERNAL 内部形式, デバイス

ASK SCREEN文 TB

書式 ASK SCREEN〈数値変数1〉,〈数値変数2〉,〈数値変数3〉,〈数値変数4〉

機能 カレント・ウィンドウの左端, 右端, 上端, 下端を, スクリーン座標で〈数値変数1〉〈数値変数2〉〈数値変数3〉〈数値変数4〉に代入します。

ASK SETTER文 TB

書式 ASK #〈整数式〉: SETTER〈数値変数〉

機能 ファイルのレコード・ポインタが移動できるかどうかを〈文字列変数式〉に代入します。
YES 可変長ファイル, 固定長ファイル

ASK TEXT JUSTIFY文 TB

書式 ASK TEXT JUSTIFY〈文字列変数式1〉,〈文字列変数式2〉

機能 文字列の水平と垂直の表示位置を〈文字列変数式1〉〈文字列変数式2〉に代入します。
〈文字列変数式1〉には, 水平の位置を代入します。
LEFT 左端 (デフォルト値)
RIGHT 右端
CENTER 中央

〈文字列変数式2〉には垂直の位置を代入します。
TOP 上端
BOTTOM 下端
BASE 基線 (デフォルト値)
HALF 中央

ASK WINDOW文 TB

書式 ASK WINDOW〈数値変数1〉,〈数値変数2〉,〈数値変数3〉,〈数値変数4〉

機能 カレンド・ウィンドウの左端, 右端, 上端, 下端を, ウィンドウ座標で〈数値変数1〉〈数値変数2〉〈数値変数3〉〈数値変数4〉に代入します。

ASK ZONEWIDTH文 TB

書式 ASK ZONEWIDTH〈数値変数〉
ASK #〈整数式〉: ZONEWIDTH〈数値変数〉

機能 ウィンドウやファイルの領域幅を〈数値変数〉に代入します。

ATN関数 N88/TB/QB

書式 ATN (〈数値式〉)

機能 〈数値式〉のアークタンジェント (逆正接) の値を返します。

ATTR$ (DISKモード) N88

書式 ATTR$ (〈ドライブ番号〉)
#〈ファイル番号〉
〈ファイル・ディスクリプタ〉

機能 ファイル, ドライブの属性を得ます。

AUTO N88

書式 AUTO [〈行番号〉] [, 〈増分〉]

機能 行番号を自動的に発生します。

BEEP N88/QB

書式 BEEP [〈スイッチ〉]

機能 内蔵スピーカを鳴らしたり, 止めたりします。

BLOAD (DISKモード) N88

書式 BLOAD〈ファイル・ディスクリプタ〉[, 〈ロードアドレス〉] [, R]

機能 機械語ファイルをメモリ上にロードします。

BLOAD QB

書式 BLOAD *filespec* [, *offset*]

機能 BSAVEステートメントで作ったメモリ・イメージファイルを, 入力ファイルまたはデバイスからメモリに読み込みます。

BOX AREA文 TB

書式 BOX AREA〈ボックス座標〉
〈ボックス座標〉: :〈数値式1〉,〈数値式2〉,〈数値式3〉,〈数値式4〉

機能 〈ボックス座標〉で指定した長方形を描き, その内部を塗りつぶします。

BOX CIRCLE文 TB

書式 BOX CIRCLE〈ボックス座標〉
〈ボックス座標〉: :〈数値式1〉,〈数値式2〉,〈数値式3〉,〈数値式4〉

機能 〈ボックス座標〉で指定した長方形に内接する楕円を描きます。

BOX CLEAR文 TB

書式 BOX CLEAR〈ボックス座標〉
〈ボックス座標〉: :〈数値式1〉,〈数値式2〉,〈数値式3〉,〈数値式4〉

機能 〈ボックス座標〉で指定した長方形の領域を消去します。

BOX ELLIPSE文 TB

機能 BOX CIRCLE文と同じです。

BOX KEEP文 TB

書式 BOX KEEP〈ボックス座標〉IN〈文字列変数式〉

〈ボックス座標〉：：〈数値式１〉，〈数値式２〉，〈数値式３〉，〈数値式４〉
〈文字列変数式〉：：〈文字列変数〉

機能　〈文字列変数〉〈部分文字列式〉〈ボックス座標〉で指定した長方形の領域全体を，画素の形式で〈文字列変数式〉に格納します。

BOX LINES文　TB

書式　BOX LINES〈ボックス座標〉
〈ボックス座標〉：：〈数値式１〉，〈数値式２〉，〈数値式３〉，〈数値式４〉

機能　〈ボックス座標〉で指定した長方形の枠を描きます。

BOX SHOW文　TB

書式　BOX SHOW〈文字列式〉AT〈数値式１〉，〈数値式２〉
BOX SHOW〈文字列式〉AT〈数値式１〉，〈数値式２〉USING〈表示条件〉
BOX SHOW〈文字列式〉AT〈数値式１〉，〈数値式２〉USING〈整数式〉
〈表示条件〉：：”AND”
”OR”
”XOR”

文字列に格納されたイメージを長方形の部分に復元します。長方形の左下端は，〈数値式１〉〈数値式２〉で指定します。

BSAVE (DISKモード)　N88

書式　BSAVE〈ファイル・ディスクリプタ〉，〈開始アドレス〉，〈長さ〉

機能　メモリ上の指定範囲の内容を，ディスク上あるいはRS-232C回線上に機械語ファイルとしてセーブします。

BSAVEステートメント　QB

書式　BSAVE *filespec, offset, length*

機能　メモリ領域の内容を，出力ファイルまたはデバイスに転送します。

BREAK文　TB

書式　BREAK

機能　デバッグが実行状態の場合に，エラーを発生させます。

BREAK　TB

書式　BREAK
BREAK Myfunc

機能　［引数なし］カーソルがある行にブレーク・ポイントを付けます。
［引数一つ］指定した行にブレーク・ポイントを付けます。

BYE　TB

書式　BYE

機能　［引数なし］True BASICを終了します。

CALL　N88

書式　CALL〈変数名〉〔(〈引数〉〔,〈引数〉…〕)〕

機能　メモリ上に用意された機械語サブルーチンを呼び出し，実行します。

CALL文　TB

書式　CALL〈識別子〉
CALL〈識別子〉(〈サブルーチン引数リスト〉)
〈サブルーチン引数リスト〉：：〈サブルーチン引数〉．．．．〈サブルーチン引数〉
〈サブルーチン引数〉：：〈数値式〉
〈文字列式〉
〈配列引数〉
＃〈整数式〉

機能　〈識別子〉で指定したサブルーチンを呼び出します。〈サブルーチン引数リスト〉は，SUB文のパラメータと一致していなければなりません。

CALLステートメントーBASICのプロシージャ　QB

書式　1 CALL*name* [(*argumentlist*)]
2 name [argumentlist]

機能　制御をQuickBASICのSUBプロシージャに移します。

CALL, CALLSステートメントー他の言語のプロシージャ　QB

書式　1 CALL*name* [(*call-argumentlist*)]
2 name [call-argumentlist]
3 CALLS*name* [(*calls-argumentlist*)]

機能　制御を他の言語で書いたプロシージャに移します。

CALL ABSOLUTE　QB

書式　CALL ABSOLUTE*name*([*argumentlist*,] *integervariable*)

機能　制御をマシン語のプロシージャに移します。

CALL INT86OLD　QB

書式　1 CALL INT86OLD(*int no, in array*( ), *out array*( ))
2 CALL INT86XOLD(*int no, in array*( ), *out array*( ))

機能　プログラムからDOSのシステムコールを呼び出します。

CALL INTERRUPT　QB

書式　1 CALL INTERRUPT(*interruptnum, inregs, outregs*)
2 CALL INTERRUPTX(*interruptnum, inregs, outregs*)

機能　プログラムからDOSのシステム・コールを呼び出します。

CAUSE文　TB

書式　CAUSE EXCEPTION〈整数式〉
CAUSE EXCEPTION〈整数式〉，〈文字列式〉
CAUSE ERROR〈整数式〉
CAUSE ERROR〈整数式〉，〈文字列式〉

機能　実行時エラーを定義します。〈整数式〉をエラー番号〈文字列式〉をエラー・メッセージにします。

CD　TB

書式　CD SUBDIR
CD B:¥TRUE

機能　［引数一つ］カレント・ディレクトリを変更します。

CDBL　N88/QB

書式　CDBL (〈数式〉)

機能　整数値，単精度実数値を，倍精度実数値に変換します。

CEIL関数　TB

書式　CEIL〈数値式〉

機能　〈数値式〉の値以上の最小の整数を返します。

CHAIN (DISKモード)　N88

書式　CHAIN〔MERGE〕〈ファイル・ディスクリプタ〉〔,〈行番号〉〕〔,CALL〕〔,DELETE〈範囲〉〕

機能　メモリ上のプログラムからディスク上のプログラムに実行を移します。

CHAIN文　TB

書式　CHAIN〈文字列式〉
CHAIN〈文字列式〉WITH (〈関数引数リスト〉)
CHAIN〈文字列式〉，RETURN
CHAIN〈文字列式〉WITH (〈関数引数リスト〉)，RETURN
〈関数引数リスト〉：：〈引数〉．．．．〈引数〉
〈引数〉：：〈数値式〉
〈文字列式〉
〈数値変数〉
〈文字列配列〉

機能　カレント・プログラムを停止して，〈文字列式〉で指定したプログラムを開始します。

CHANGE　TB

機能　［引数二つ］カレント・プログラムの文字列を一括して置換します。

CDBL$関数　QB

書式　CDBL$(*stringexpression*)

機能　文字列中の１バイト文字を，２バイト文字に変換します。

CHAINステートメント　QB

書式　CHAIN *filespec*

機能　制御を現在のプログラムから別のプログラムに移します。

CHDIRステートメント　QB

書式　CHDIR *pathspec*

機能　現在のデフォルトのディレクトリを，指定したディレクトリに変更します。

CHR$　N88/TB/QB

書式　CHR$（〈数式〉）

機能　指定したキャラクタ・コードを持つ文字を得ます。

CINT　N88/QB

書式　CINT（〈数式〉）

機能　単精度実数値，倍精度実数値を，整数倍に変更します。

CIRCLE　N88

書式　CIRCLE |(Wx,Wy)|,〈半径〉〔,〈パレット番号1〉〕
　　　　　　|STEP (x,y)|
〔,〈開始角度〉〕〔,〈終了角度〉〕〔,〈比率〉〕〔, F〔,
　|〈パレット番号2〉|〕〕
　|〈タイル・ストリング〉|

機能　円，楕円を描きます。

CIRCLEステートメント　QB

書式　CIRCLE [STEP] (*x*, *y*), *radius* [, [*colar*] [, [*start*] [, [*end*] [, *aspect*]]]]

機能　指定した中心と半径を持つ楕円や円を描きます。

CLEAR　N88

書式　CLEAR〔〈ダミー・パラメータ〉〕〔,〈メモリの上弦〉〕〔,〈スタックの大きさ〉〕〔,〈配列データ領域の大きさ〉〕

機能　変数の初期化およびメモリ・レイアウトを決定します。

CLEAR文　TB

書式　CLEAR

機能　カレント・ウィンドウを消去します。

CLEARステートメント　QB

書式　CLEAR [, , *stack*]

機能　すべてのプログラム変数を初期化し，ファイルを閉じて，スタック・サイズを設定し直します。

CLNG関数　QB

書式　CLNG(*numeric-expression*)

機能　小数部を丸めて，指定した数式を長整数（4バイト）に変換します。

CLOSE　N88

書式　CLOSE〔〔#〕〈ファイル番号〉〔,〔#〕〈ファイル番号〉〕…〕

機能　ファイルを閉じます。

CLOSE　TB

使用例　CLOSE
　　　　CLOSE HANOI

機能　［引数なし］カレント・プログラムをクローズします。
　　　［引数一つ］オープンしているプログラムをクローズします。

CLOSE文　TB

書式　CLOSE #〈整数式〉

機能　ファイルやウィンドウをクローズします。

CLOSEステートメント　QB

書式　CLOSE [[#] *filenumber* [, [#] *filenumber*]…]

機能　ファイルやデバイスへの入出力を終了します。

CLS　N88/QB

書式　CLS〔〈機能〉〕

機能　現在アクティブな画面をクリアします。

CMD BREAK　N88

書式　CMD BREAK〔〔#〕〈電話機番号〉,〕〈時間〉
　　　CMD BREAK〔〔#〕〈電話機番号〉〕ON
　　　CMD BREAK〔〔#〕〈電話機番号〉〕OFF

機能　ブレイク信号の送出を制御します。

CMD CHANGE DUPLEX　N88

書式　CMD CHANGE DUPLEX 〈ポート番号〉〔,〈通信方式〉〕

機能　通信方式（全二重／半二重方式）を切り換えます。

CMD DIAL　N88

書式　1) CMD DIAL 〔〔#〕〈電話機番号〉,〕〈電話番号〉〔,〈機能〉〕
　　　2) CMD DIAL 〔〔#〕〈電話機番号〉,〕〈短縮番号〉

機能　電話をかけます。

CMD ERROR ON/OFF/STOP　N88

書式　1) CMD ERROR 〔〔#〕〈電話機番号〉〕ON
　　　2) CMD ERROR 〔〔#〕〈電話機番号〉〕OFF
　　　3) CMD ERROR 〔〔#〕〈電話機番号〉〕STOP

機能　通信エラーによる割り込みを許可，禁止，停止します。

CMD LINE CLOSE　N88

書式　CMD LINE CLOSE 〔〔#〕〈電話機番号〉〕

機能　BASICと論理的に接続されている電話機を切り離します。

CMD LINE ON/OFF/STOP　N88

書式　1) CMD LINE 〔〔#〕〈電話機番号〉〕ON
　　　2) CMD LINE 〔〔#〕〈電話機番号〉〕OFF
　　　3) CMD LINE 〔〔#〕〈電話機番号〉〕STOP

機能　電話機の着信による割り込みを許可，禁止，停止します。

CMD LINE OPEN　N88

書式　CMD LINE OPEN〈ファイル・ディスクリプタ〉〔AS〔#〕〈電話機番号〉〕

機能　電話機をBASICと論理的に接続し，オートダイヤル，自動発着信などの機能を利用可能にします。

CMD MODE CUT　N88

書式　CMD MODE CUT〔〔#〕〈電話機番号〉〕

機能　電話を切ります。

CMD ON ERROR GOSUB　N88

書式　CMD ON ERROR 〔〔#〕〈電話機番号〉〕GOSUB〈行番号〉

機能　通信エラーによる割り込みルーチンの開始行を定義します。

CMD ON LINE GOSUB　N88

書式　CMD ON LINE 〔〔#〕〈電話機番号〉〕GOSUB〈行番号〉

機能　電話機の着信による割り込みが発生したときの処理ルーチンの開始行を定義します。

CMD RECEIVE　N88

書式　CMD RECEIVE〔〔#〕〈電話機番号〉,〕〔,〈機能〉〕

機能　電話機の自動着信を行なうか否かの設定をします。

CMD STORE DIAL　N88

書式　CMD STORE DIAL〔〔#〕〈電話機番号〉,〕〈短縮番号〉AS〈電話番号〉〔,〈機能〉〕

機能　短縮ダイヤルを電話機に記憶させます。

(1) COLOR　N88/QB

書式　COLOR〔〈ファンクション・コード〉〕〔,〈バックグラウンド・カラー〕〔,〈ボーダー・カラー〉〕〔,〈フォアグラウンド・カラー〉〕〔,〈パレット・モード〉〕

機能　ディスプレイ画面の各部の色およびグラフィック画面のパレット・モードを指定します。

(2) COLOR　N88

書式　COLOR〔=(〈パレット番号〉,〈カラーコード〉)〕
機能　カラー・パレットの色を変更します。

COLOR@　N88

書式　COLOR@　(X1, Y1)－(X2, Y2)〔,〈ファンクション・コード〉〕
機能　テキスト画面に書かれた文字などに色や機能を設定します。

COLORS　TB

使用例　COLORS to Cfile
COLORS from Cfile
機能　［引数一つ］エディタの各部分の色設定を保存したり，呼び出したりします。

COM ON/OFF/STOP　N88/QB

書式　1)COM　〔(〈回線番号〉)〕ON
2)COM　〔(〈回線番号〉)〕OFF
3)COM　〔(〈回線番号〉)〕STOP
機能　RS-232C回線からの割り込みの許可，禁止，停止を制御します。

COMMAND$関数　QB

書式　COMMAND$
機能　プログラムを起動したときに,コマンド・ライン上で指定された引数リストを返します。

COMMONステートメント　QB

書式　COMMON［SHARED］［/*blockname*/］*variablelist*
機能　モジュール間で共有したり，別のプログラムに引き渡すグローバル変数を定義します。

COMMON (DISKモード)　N88

書式　COMMON〈変数名〉〔,〈変数名〉…〕
機能　CHAINが実行された際，メモリ上のプログラムから，実行の移されたプログラムに変数を引き渡します。

COMPILE　TB

書式　COMPILE
機能　［引数なし］カレント・プログラムをコンパイルします。

CON配列定数　TB

書式　CON〈添字変更式〉
CON
機能　要素がすべて1である数値配列を返します。CONが使えるのは，MAT代入中の文だけです。〈添字変更式〉があるときは，その〈添字変更式〉で指定した次元の配列が生成されます。〈添字変更式〉がないときは，元の配列の次元のままです。

CONSOLE　N88

書式　CONSOLE　〔〈スクロール開始行〉〕〔,〈スクロール行数〉〕〔,〈ファンクション・キー表示スイッチ〉〕〔,〈カラー／白黒スイッチ〉〕
機能　テキスト画面モードの設定を行ないます。

CONSTステートメント　QB

書式　CONST *constantname* = *expression*　〔, *constantname* = *expression*〕...
機能　数値や文字列の代りに使う記号定数を宣言します。

CONT　N88

書式　CONT
機能　STOPキーあるいはCTRL＋Cの入力，またはSTOPによって停止したプログラムの実行を再開します。

CONTINUE　TB

使用例　CONTINUE
機能　［引数なし］停止しているカレント・プログラムの実行を継続します。

CONTINUE文　TB

書式　CONTINUE
機能　エラーの原因になった文に続く文に移動します。WHEN構文かHANDLER構文のハンドラ部でだけ使えます。

COPY　TB

使用例　COPY
COPY Mysub, Myfunc
機能　［引数なし］マークが付いているカーソルのある行い以降に複写します。
［引数二つ］行のまとまりを，指定した行に複写します。

COPY　N88

書式　COPY　〔〈機能〉〕
機能　画面情報のハードコピーを行ないます。

COS関数　TB

書式　COS　(〈数値式〉)
機能　〈数値式〉が示す角度のコサイン(余弦)の値を返します。

COS　N88/QB

書式　COS　(〈数式〉)
機能　余弦（コサイン）を得ます。

COSH関数　TB

書式　COSH　(〈数値式〉)
機能　〈数値式〉の双曲線コサインの値を返します。

COT関数　TB

書式　COT　(〈数値式〉)
機能　〈数値式〉が示す角度のコタンジェント（余接）の値を返します。

CPOS関数　TB

書式　CPOS　(〈文字列式1〉,〈文字列式2〉)
CPOS　(〈文字列式1〉,〈文字列式2〉,〈整数式〉)
機能　〈文字列式2〉内のいずれかの文字が，〈文字列式1〉で最初に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，右へ進みます。

CPOSR関数　TB

書式　CPOSR　(〈文字列式1〉,〈文字列式2〉)
CPOSR　(〈文字列式1〉,〈文字列式2〉,〈整数式〉)
機能　〈文字列式2のいずれかの文字が，〈文字列式1〉で最後に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，左へ進みます。

CSC関数　TB

書式　CSC　(〈数値式〉)
機能　〈数値式〉が示す角度のコセカント（余剰）の値を返します。

CSNG　N88/QB

書式　CSNG　(〈数値式〉)
機能　整数値，倍精度実数値を，単精度実数値に変換します。

CSNG$関数　QB

書式　CSNG$(*stringexpression*)
機能　文字列中の2バイト文字を，1文字バイト文字に変換します。

CSRLIN　N88

書式　CSRLIN
機能　現在のカーソルの行位置を得ます。

CVI/CVS/CVD　N88

書式　CVI　(〈2文字の文字列〉)
CVS　(〈4文字の文字列〉

CVD (〈8文字の文字列〉)

機能　文字列を数値データに変換します。

CVI/CVS/CVL/CVD関数　QB

書式　CVI(2-*byte-string*)
CVS(4-*byte-string*)
CVL(4-*byte-string*)
CVD(8-*byte-string*)

機能　数値を含む文字列を数値に変換します。

CVSMBF, CVDMBF関数　QB

書式　CVSMBF(4-*byte-string*)
CVDMBF(8-*byte-string*)

機能　Microsoftバイナリ形式の数値を含む文字列を，IEEE形式の数値に変換します。

DATA　QB

書式　DATA 〈定数〉〔,〈定数〉...〕

機能　READで読み込まれる数値定数，文字定数を定義します。

DATA文　TB

書式　DATA 〈データ〉....〈データ〉
〈データ〉::〈ダブル・クォーテーション・マークで囲まれた文字列〉
〈ダブル・クォーテーション・マークで囲まれていない文字列〉

機能　プログラムを実行したときに，〈データ〉を順にデータリストに格納します。

DATE関数　TB

書式　DATE

機能　現在の日付を，十進法の数を使ってYYDDDという形式で返します。引数はありません。YYは西暦年で，DDDはその年の通算の日付の番号です。

DATE$　N88

書式　1) DATE$
2) DATE$="yy/mm/dd"

機能　日付を得ます。

DATE$関数　TB

書式　DATE$

機能　現在の日付を，文字列を使ってYYYYMMDDという形式で返します。引数はありません。YYYYが西暦年，MMが月，DDが日を示します。

DATE$ステートメント　QB

書式　DATE$=*stringexpression*

機能　現在の日付を設定します。

DEBUG文　TB

書式　DEBUG ON
DEBUG OFF

機能　DEBUG ONを使うとデバッグがはじまります。DEBUG OFF文を使うとデバッグが終了します。

DECLAREステートメント－BASICのプロシージャ　QB

書式　DECLARE {FUNCTION | SUB} *name* [([*parameter list*])]

機能　BASICのプロシージャを宣言し，引数のデータ型をチェックするようコンパイラに指示します。

DECLAREステートメント－他の言語のプロシージャ　QB

書式　1 DECLARE FUNCTION *name* [CDECL] [ALIAS"*aliasname*"] [([*parameterlist*])]
2 DECLARE SUB*name* [CDECL] [ALIAS"*aliasname*"] [([*parameterlist*])]

機能　他の言語で書いた外部プロシージャを呼び出す手順を宣言します。

DECLARE DEF文　TB

機能　DECLARE FUNCTION文と同じです。

DECLARE FUNCTION文　TB

書式　DECLARE FUNCTION 〈関数名〉....〈関数名〉
DECLARE INTERNAL FUNCTION 〈関数名〉....〈関数名〉
DECLARE EXTERNAL FUNCTION 〈関数名〉....〈関数名〉
〈関数名〉::〈識別子〉
〈文字識別子〉

機能　ユーザ定義関数を宣言します。

DECLARE NUMERIC文　TB

書式　DECLARE NUMERIC 〈数値宣言〉....〈数値宣言〉
〈数値宣言〉::〈単一数値変数〉
〈数値配列名〉〈添字範囲〉

機能　数値変数または数値配列を宣言します。

DECLARE PUBLIC文　TB

書式　DECLARE PUBLIC 〈公用名〉
〈公用名〉::〈単一数値変数〉
〈単一文字列変数〉
〈配列パラメータ〉

機能　公用変数を宣言します。

DECLARE STRING文　TB

書式　DECLARE STRING 〈文字列宣言〉....〈文字列宣言〉
DECLARE STRING 〈最大長〉〈文字列宣言〉....〈文字列宣言〉
〈文字列宣言〉::〈単一文字列変数〉
〈単一文字列変数〉〈最大長〉
〈文字列配列名〉〈添字範囲〉
〈文字列配列名〉〈添字範囲〉〈最大長〉
〈最大長〉::＊〈整数〉

機能　文字列変数又は文字列配列を宣言します。〈最大長〉があれば，その文字列変数や文字列配列の最大長になります。

DECLARE SUB文　TB

書式　DECLARE SUB 〈サブルーチン名〉....〈サブルーチン名〉
DECLARE INTERNAL SUB〈サブルーチン名〉....〈サブルーチン名〉
DECLARE EXTERNAL SUB〈サブルーチン名〉....〈サブルーチン名〉
〈サブルーチン名〉::〈識別子〉

機能　現在のTrue BASICのバージョンでは無効ですが，ANSI規格との互換性を保つためにあります。

DEF文　TB

書式　FUNCTION文と同じです。

DEF構文　TB

書式　FUNCTION構文と同じです。

DEF FN　N88/QB

書式　DEF FN 〈名前〉〔(〈パラメータ・リスト〉)〕=〈関数の定義式〉

機能　利用者定義関数を指定します。

DEFINT/DEFSNG/DEFDBL/DEFSTR　N88

書式　1) DEFINT 〈文字の範囲〉〔,〈文字の範囲〉…〕
2) DEFSNG 〈文字の範囲〉〔,〈文字の範囲〉…〕
3) DEFDBL 〈文字の範囲〉〔,〈文字の範囲〉…〕
4) DEFSTR 〈文字の範囲〉〔,〈文字の範囲〉…〕

機能　変数の型宣言を行ないます。

DEF SEGステートメント　QB

書式　DEF SEG [=*address*]

機能　後に続くPEEK関数，BLOAD，BSAVE，CALL AB

SOLUTE，POKEステートメントで使う，セグメント・アドレスを設定します。

DEF SEG　N88

書式　DEF SEG＝〈セグメント・ベース〉

機能　セグメント・ベースを宣言します。

DEFtypeステートメント　QB

書式　DEFINT*letterrange*［，*letterrange*］
DEFSNG*letterrange*［，*letterrange*］
DEFDBL*letterrange*［，*letterrange*］
DEFLMG*letterrange*［，*letterrange*］
DEFSTR*letterrange*［，*letterrange*］

機能　変数，DEF FN関数，FUNCTIONプロシージャのデフォルトのデータ型を設定します。

DEF USR　N88

書式　DEF USR〔〈番号〉〕＝〈開始アドレス〉

機能　USRで呼び出す機械語関数の番号と実行開始アドレスを定義します。

DEG関数　TB

書式　DEG（〈数値式〉）

機能　ラジアンで与えられた〈数値式〉を，度数に変換します。

DELETE　N88

書式　DELETE　|〈始点行番号〉〔－〈終点行番号〉〕|
|〔〈始点行番号〉〕－〈終点行番号〉|

機能　プログラムの部分削除を行ないます。

DELETE　TB

使用例　DELETE 10-30

機能　［引数一つ］行のまとまりを削除します。

DET関数　TB

書式　DET（〈数値配列〉）
DET

機能　〈数値配列〉で表される正方行列の行列式の値を返します。引数がないときは，INV関数でもっとも新しく逆行列を求めた正方行列の行列式の値を返します。

DIM文　TB

書式　DIM〈配列項目〉．．．．〈配列項目〉
〈配列項目〉：：〈数値配列〉〈添字範囲〉
〈文字列配列〉〈添字範囲〉

機能　配列の次元を定義します。

DIM　N88

書式　DIM〈変数名〉（〈添字の最大値〉〔，〈添字の最大値〉…〕）〔，〈変数名〉（〈添字の最大値〉〔，〈添字の最大値〉…〕）…〕

機能　配列変数の要素の大きさを指定し，メモリ領域に割り当てます。

DIMステートメント　QB

書式　DIM［SHARED］*variable*［（*subscripts*）］［AS*type*］［，*variable*［（*subscripts*）］［AS*type*］］．．．

機能　変数を宣言し，メモリを割り当てます。

DIVIDEサブルーチン　TB

書式　CALL DIVIDE（〈数値式1〉，〈数値式2〉，〈数値変数1〉，〈数値変数2〉）

機能　〈数値式1〉を〈数値式2〉でわって，その値を〈数値変数1〉に代入し，剰余を〈数値変数2〉に代入して返します。

DO　TB

使用例　DO format

機能　［引数一つ］プリプロセッサを実行します。

DOループ　TB/QB

書式　〈doループ〉：：〈do文〉
．．．
〈loop文〉
〈do文〉：：DO
DO WHILE〈論理式〉
DO UNTIL〈論理式〉
〈loop文〉：：LOOP
LOOP WHILE〈論理式〉
LOOP UNTIL〈論理式〉

機能　ループをつくります。WHILE条件は「真」のときにループを続け，UNTIL条件は「偽」のときにループを続けます。

DOT関数　TB

書式　DOT（〈配列引数1〉，〈配列引数2〉）

機能　〈配列引数1〉〈配列引数2〉で表される2個の配列の内積を計算して返します。

DRAW文　TB

書式　DRAW〈識別子〉
DRAW〈識別子〉（〈サブルーチン引数リスト〉）
DRAW〈識別子〉WITH（変換形式）
DRAW〈識別子〉（〈サブルーチン引数リスト〉）WITH（変換形式）
〈サブルーチン引数リスト〉：：〈サブルーチン引数〉．．．．〈サブルーチン引数〉
〈サブルーチン引数〉：：〈数値式〉
〈文字列式〉
〈配列引数〉
＃〈整数式〉
〈変換形式〉：：〈変換項目〉．．．＊〈変換項目〉
〈変換項目〉：：SCALE（〈数値式〉）
SCALE（〈数値式1〉，〈数値式2〉）
ROTATE（〈数値式〉）
SHIFT（〈数値式1〉，〈数値式2〉）
SHEAR（〈数値式〉）
〈数値配列〉

機能　〈識別子〉で指定したピクチャを描きます。〈サブルーチン引数リスト〉はPICTURE文のパラメータと一致していなければなりません。WITH句があると，指定された変換が行なわれます。変換の種類は，移動（SHIFT），回転（ROTATE），傾き（SHEAR），拡縮（SCALE）です。

DRAWステートメント　QB

書式　DRAW*stringexpression*

機能　引数*stringexpression*で指定したグラフィックス・マクロコマンドに従い，図形を描きます。

DRAW（DISKモード）　N88

書式　DRAW〈文字列式〉

機能　グラフィック描画サブ・コマンド列に従って，ワールド座標上で図形を描きます。

DSKF（DISKモード）　N88

書式　DSKF（〈ドライブ番号〉〔，〈機能〉〕

機能　ディスクに関する情報を得ます。

DSKI$（DISKモード）　N88

書式　DSKI$（〈ドライブ番号〉，〈サーフェス番号〉，〈トラック番号〉，〈セクタ番号〉）

機能　ディスクから直接，データを読み出します。

DSKO$（DISKモード）　N88

書式　DSKO$（〈ドライブ番号〉，〈サーフェス番号〉，〈トラック番号〉，〈セクタ番号〉）

機能　ディスクに対して直接書き込みを行ないます。

ECHO　TB

使用例　ECHO
ECHO to Myoutput

機能　［引数なし］コマンドの実行結果を印刷します。
［引数一つ］コマンドの実行結果をファイルに出力します。

EDIT　TB

使用例　EDIT
EDIT Yesno

機　能　［引数なし］編集操作を，マークが付いている行だけに限定します。
［引数一つ］編集結果を，指定した行のまとまりに限定します。

EDIT　N88

書　式　EDIT〈行番号〉

機　能　指定された行を画面上に表示し，以降 ROLL UP キー，ROLL DOWN キーなどによるプログラム編集を可能にします。

ENDステートメント　QB

書　式　END [{DEF | FUNCTION | IF | SELECT | SUB | TYPE}]

機　能　BASICのプログラム，プロシージャ，ステートメント・ブロックの実行を終了します。

END　N88

書　式　END

機　能　プログラムの終了を宣言します。

END文　TB

書　式　END

機　能　メイン・プログラムの最後の文です。

ENTER　TB

機　能　CDコマンドと同じです。

ENVIRON$関数　QB

書　式　1 ENVIRON$(*environmentstring*)
2 ENVIRON$(*n*)

機　能　DOSの環境文字列テーブルが指定した環境文字列を検索します。

ENVIRONステートメント　QB

書　式　ENVIRON*stringexpression*

機　能　DOSの環境文字列テーブル内のパラメータを修正します。

EOF　N88/QB

書　式　EOF（〈ファイル番号〉）

機　能　ファイルの終了コードを調べます。

EPS関数　TB

書　式　EPS（〈数値式〉）

機　能　〈数値式〉に加算したり〈数値式〉から減算したりしたときに，扱うことのできる最小の正の数を返します。

ERASE　N88

書　式　ERASE〈配列変数名〉〔,〈配列変数名〉…〕

機　能　配列変数を消去します。

ERASEステートメント　QB

書　式　ERASE*arrayname* [, *arrayname*...]

機　能　STATIC配列の要素を再初期化したり，DYNAMIC配列に割り当てたメモリを解放します。

ERASE文　TB

書　式　ERASE#〈整数式〉
ERASE REST#〈整数式〉

機　能　ファイルの内容を消去します。ERASE REST文の場合は，カレント項目からファイルの最後まで消去します。

ERDEV,ERDEV$関数　QB

書　式　ERDEV
ERDEV$

機　能　エラーが起きた後で，デバイスの状態を知らせます。

ERL/ERR　QB

書　式　1）ERL
2）ERR

機　能　エラーの発生した行番号および発生したエラーのコードを保持しています。

ERROR　N88/QB

書　式　ERROR〈整数表記〉

機　能　エラー発生のシミュレート，エラーコードのユーザー定義を行ないます。

EXITステートメント　QB

書　式　EXIT {DEF | DO | FOR | FUNCTION | SUB}

機　能　DEF FN関数，DO...LOOPループ，FOR...NEXTループ，SUBプロシージャ，FUNCTIONプロシージャを抜け出ます。

EXLINE関数　TB

書　式　EXLINE

機　能　プログラム内でもっとも新しくエラーが発生した行の番号（先頭からの行数）を返します。引数はありません。

EXLINE$関数　TB

書　式　EXLINE$

機　能　プログラム内でもっとも新しくエラーが発生した位置を，文字列にして返します。引数はありません。

EXP関数　N88/TB/QB

書　式　EXP（〈数値式〉）

機　能　自然対数の底 e の〈数値式〉乗を返します。

EXTERNAL文　TB

書　式　EXTERNAL

機　能　ライブラリの中のプロシージャを外部プロシージャに指定します。

EXTEXT$関数　TB

書　式　EXTEXT$

機　能　エラーがカラー・ハンドラによってチェックされた場合に，もっとも新しく発生したエラーかCAUSE EXCEPTION文に関連するエラー・メッセージの文字列を返します。引数はありません。

EXTYPE関数　TB

書　式　EXTYPE

機　能　エラーがカラー・ハンドラによってチェックされた場合に，もっとも新しく発生したエラーの番号を返します。引数はありません。

FIELD (DISKモード)　N88/QB

書　式　FIELD〔#〕〈ファイル番号〉,〈フィールド幅〉AS〈文字変数〉〔,〈フィールド幅〉AS〈文字変数〉…〕

機　能　ランダム・ファイル・バッファにフィールド変数を割り当てます。

FILEATTR関数　N88

書　式　FILEATTR(*filenumber*, *attribute*)

機　能　オープンしているファイルの情報を返します。

FILES　TB

使用例　FILES
FILES *.*

機　能　［引数なし］拡張子が.TRU，TRCのファイルのリストを表示します。
［引数一つ］指定したファイルのリストを表示します。

FILES/LFILES (DISKモード)　N88

書　式　1）FILES〔〈ドライブ番号〉〕
2）LFILES〔〈ドライブ番号〉〕

機　能　ディスクに入っているファイルの名前，種類，大きさを出力

します。

FILESステートメント　QB

書式　FILES ［*filespec*］
機能　指定したディスク上にあるファイルの一覧を画面に表示。

FIND　TB

使用例　FIND
FIND COLOR
機能　［引数なし］検索する文字列を指定します。
［引数一つ］指定した文字列を検索します。

FIX　N88/QB

書式　FIX(〈数式〉)
機能　数値の整数部を得ます。

FLOOD文　TB

書式　FLOOD〈数値式1〉，〈数値式2〉
機能　〈数値式1〉〈数値式2〉で指定した点を含む閉じた領域を塗りつぶします。

FOR…TO…STEP～NEXT　N88/QB

書式　FOR　〈変数名〉=〈初期値〉 TO　〈終値〉〔STEP〈増分〉〕
⌇
NEXT　〔〈変数名〉〕〔，〈変数名〉…〕
機能　FORからNEXTまでの中間にある一連の命令を繰り返して実行。

FORループ　TB

書式　〈forループ〉：：〈for文〉
...
NEXT〈単一数値変数〉
〈for文〉：：FOR〈単一数値変数〉=〈数値式1〉TO〈数値式2〉
FOR〈単一数値変数〉=〈数値式1〉TO〈数値式2〉
STEP
〈数値式3〉
機能　指定した回数だけ繰り返して実行します。〈数値式1〉が初期値，〈数値式2〉が終了値，〈数値式3〉が増分です。

FORGET　TB

使用例　FORGET
機能　［引数なし］メモリを解放します。

FP関数　TB

書式　FP　(〈数値式〉)
機能　〈数値式〉の値の小数部分を返します。

FPOS　N88

書式　FPOS　(〈ファイル番号〉)
機能　ファイル中での物理的な現在位置を示します。

FRE　N88/QB

書式　FRE　(〈機能〉)
機能　メモリの未使用領域の大きさを得ます。

FREEFILE関数　QB

書式　FREEFILE
機能　次に使用可能なファイル番号を返します。

FUNCTIONステートメント　QB

書式　FUNCTION*name*　［(*parameterlist*)］［STATIC］
⌇
name=expression
⌇
NED FUNCTION
機能　FUNCTIONプロシージャを構成する，名前，パラメータ，コードを宣言します。

FUNCTION文　TB

書式　FUNCTION〈識別子〉=〈数値式〉
FUNCTION〈識別子〉(〈関数定義パラメータ〉....〈関数定義パラメータ〉)=〈数値式〉
FUNCTION〈文字列識別子〉=〈文字列式〉
FUNCTION〈文字列識別子〉(〈関数定義パラメータ〉....〈関数定義パラメータ〉)=〈数値式〉
〈関数定義パラメータ〉：：〈単一数値変数〉
〈単一文字列変数〉
〈配列パラメータ〉
機能　ユーザ定義関数を1行で定義します。

FUNCTION構文　TB

書式　〈関数定義構文〉：：〈関数定義開始文〉
...
END FUNCTION
〈関数定義開始文〉：：FUNCTION〈識別子〉
FUNCTION〈識別子〉〈関数定義パラメータ・リスト〉
FUNCTION〈文字列識別子〉
FUNCTION〈文字列識別子〉〈関数定義パラメータ・リスト〉
〈関数定義パラメータ・リスト〉：：〈関数定義パラメータ〉....〈関数定義パラメータ〉
〈関数定義パラメータ〉：：〈単一数値変数〉
〈単一文字列変数〉
〈配列パラメータ〉
機能　ユーザ定義関数を複数行で定義します。

GET　N88

書式　GET　〔#〕〈ファイル番号〉〔，〈数値〉〕
機能　ファイル中のデータをファイル・バッファに読み込みます。

GETステートメント－ファイルI/O　QB

書式　GET ［#］ *filenumber*［，［*recordnumber*］［，*variable*］］
機能　ディスクファイルの内容を，ランダム・アクセス・ファイルのバッファや変数に読み込みます。

GETステートメント－グラフィックス　QB

書式　GET ［STEP］(*x1*,*y1*)－［STEP］(*x2*,*y2*)，*arrayname*［(*indices*)］
機能　画面上のグラフィックス・イメージを配列内に格納します。

GET@　N88

書式　GET〔@〕(Sx1，Sy1)－ ｜(Sx2，Sy2) / STEP (x，y)｜，〈配列変数名〉〔(〈添字〉)〕
機能　画面上のグラフィックス・パターンを配列変数に読み込みます。

GET KEY文　TB

書式　GET KEY〈数値変数〉
GET KEY：〈数値変数〉
機能　キーボード入力バッファ内の次の文字に対応する数値を，〈数値変数〉に代入します。

GET MOUSE文　TB

書式　GET MOUSE〈数値変数1〉，〈数値変数2〉，〈数値変数3〉
GET MOUSE：〈数値変数1〉，〈数値変数2〉，〈数値変数3〉
機能　マウスの現在位置の座標が〈数値変数1〉〈数値変数2〉に代入され，状態が〈数値変数3〉に次のように代入されます。
0 どのボタンも押されていていない
1 ボタンが押されている
2 この点でボタンがクリックされた
3 この点でボタンが解除された
4 この点でボタンがシフト・クリックされた

GET POINT文　TB

書式　GET POINT〈数値変数1〉，〈数値変数2〉
GET POINT：〈数値変数1〉，〈数値変数2〉
機能　プログラムの実行中にグラフィック・カーソルで座標を指定します。座標は〈数値変数1〉〈数値変数2〉に代入されます。

GOSUB文 N88/TB/QB

書式 GOSUB 〈行番号〉
GO SUB 〈行番号〉

機能 指定した〈行番号〉にジャンプします。対応するRETURN文が実行されるとGOSUB文の次の文に戻ります。

GOTO文 N88/TB

書式 GOTO 〈行番号〉
GO TO 〈行番号〉

機能 指定した行番号にジャンプします。

GOTOステートメント QB

書式 GOTO {*linelabel* | *linenumber*}

機能 指定した行へ無条件に分岐します。

HANDLER構文 TB

書式 HANDLER構文：：HANDLER 〈ハンドラ名〉
〈ハンドラ部〉
END HANDLER
〈ハンドラ部〉：：〈識別子〉
〈ハンドラ部〉：：... 〈文〉

機能 エラー・ハンドラのハンドラ部を分離します。エラーがWHEN EXCEPTION USE文のWHEN部で発生すると，WHEN EXCEPTION USEで宣言されたHANDLER構文に制御が移動します。

HELP TB

書式 HELP RUN

機能 ［引数一つ］コマンド，関数，文などのヘルプ画面を表示します。

HELP ON/OFF/STOP N88

書式 1)HELP ON
2)HELP OFF
3)HELP STOP

機能 HELP キーによる割り込みの許可，禁止，停止を制御します。

HEX$ N88/QB

書式 HEX$ 〈数式〉

機能 10進数を16進数に変換し，その文字列を得ます。

IDN配列定数 TB

書式 IDN 〈添字変更式〉
IDN

機能 単位行列を返します。IDNが使えるのは，MAT代入文内だけです。〈添字変更式〉があって正方行列であれば，その次元と同じものが生成されます。〈添字変更式〉で表される行列が1次元の行列である場合は，〈添字変更式〉の値を次元の数として正方行列を生成します。〈添字変更式〉がなければ，MAT代入文で代入された配列の次元と一致します。

IF文 TB/QB

書式 IF 〈論理式〉 THEN 〈単一文〉
IF 〈論理式〉 THEN 〈単一文1〉 ELSE 〈単一文2〉

機能 〈論理式〉の真偽により制御の方向を変えます。1行のIF構文です。

IF構文 TB/QB

書式 〈if構文〉：：IF 〈論理式〉 THEN
...
ELSEIF 〈論理式〉 THEN
...
ELSEIF 〈論理式〉 THEN
...
ELSE
...
END IF

機能 〈論理式〉の真偽により制御の方向を変えます。

IF…THEN～ELSE/IF…GOTO～ELSE N88

書式 1) IF 〈論理式〉 THEN {〈文〉 | 〈行番号〉} 〔ELSE {〈文〉 | 〈行番号〉}〕
2) IF 〈論理式〉 GOTO 〈行番号〉 〔ELSE {〈文〉 | 〈行番号〉}〕

機能 論理式の条件判断を行ないます。

INCLUDE TB

使用例 INCLUDE HANOI

機能 ［引数一つ］保存されているファイルの内容を，エディット・ウィンドウに挿入します。

INFO TB

使用例 INFO

機能 ［引数なし］現時点でのTrueBASICの環境についての情報を表示します。

INKEY$ N88

書式 INKEY$

機能 押されているキーの文字を得ます。

INP N88/QB

書式 INP 〈ポート番号〉

機能 入力ポートから値を読み取ります。

INPUT文 TB

書式 INPUT 〈入力リスト〉
INPUT 〈入力リスト〉,
INPUT 〈入力オプション〉 .... 〈入力オプション〉：〈入力リスト〉
INPUT# 〈整数式〉：〈入力リスト〉
INPUT# 〈整数式〉,〈入力オプション〉 .... 〈ファイル入力オプション〉：〈入力リスト〉
〈入力リスト〉：：〈変数〉 .... 〈変数〉
〈変数〉：：〈数値変数〉
〈文字列変数〉
〈文字列変数〉〈部分文字列式〉
〈ファイル入力オプション〉：：〈入力オプション〉
IF MISSING THEN 〈ジャンプ〉
〈入力オプション〉：：PROMPT 〈文字列式〉
TIMEOUT 〈数値式〉
ELAPSED 〈数値変数〉
〈ジャンプ〉：：EXIT DO
EXIT FOR
〈行番号〉
〈応答入力〉：：〈入力項目〉 .... 〈入力項目〉
〈入力項目〉 .... 〈入力項目〉,
〈入力項目〉：：〈ダブル・クォーテーション・マークで囲まれている文字列〉
〈ダブル・クォーテーション・マークで囲まれていない文字列〉

機能 キーボードやファイルから入力した〈応答入力〉を〈入力リスト〉の〈変数〉に代入します。

INPUT QB

書式 INPUT 〔〈プロンプト文〉 {; | ,}〕〈変数〉〔,〈変数〉…〕

機能 キーボードから入力されたデータを，変数に代入します。

INPUT$関数 QB

書式 INPUT$(*n* 〔, 〔#〕 *filenumber*〕)

機能 指定したファイルまたはデバイスから読み取った文字列（バイト数単位）を返します。

INPUT$ N88

書式 INPUT$ (〈文字数〉〔,〔#〕〈ファイル番号〉〕)

機能 指定されたファイルより指定された長さの文字列を読み込みます。

INPUT# N88/QB
書式 INPUT #〈ファイル番号〉, 〈変数名〉〔, 〈変数名〉…〕
機能 シーケンシャル・ファイルからデータを読み込み，変数に代入します。

INPUT¥関数 QB
書式 INPUT¥(*n*, 〔, 〔#〕 *filenumber*〕)
機能 指定したファイルまたはデバイスから読み取った文字列（文字数単位）を返します。

INPUT WAIT N88
書式 INPUT WAIT 〈待ち時間〉, 〔〈プロンプト文〉{；|，}〕〈変数名〉〔, 〈変数名〉…〕
機能 キーボードから入力されたデータを，変数に代入します。その際，入力待ち時間を制限することができます。

INSTR N88
書式 INSTR（〔〈位置〉,〕〈文字列1〉，〈文字列2〉
機能 文字列の中から指定文字列を捜して，その文字の位置を得ます。

INSTR関数 QB
書式 INSTR(〔*start*,〕 *stringexpression1*, *stringexpression2*)
機能 ある文字列の中で別の文字列を検索し，それが最初に見つかったバイト位置を返します。

INT関数 TB/QB
書式 INT（〈数値式〉）
機能 〈数値式〉の値を越えない最大の整数を返します。

INT N88
書式 INT〈数式〉
機能 小数点以下を切り捨てた整数値を得ます。

INV配列関数 TB
書式 INV（〈数値配列〉）
機能 〈数値配列〉で表される行列の逆行列を返します。〈数値配列〉で表される行列は，2次元の正方行列でなくてはなりません。INVが使えるのは，MAT代入文内だけです。

IMAGE文 TB
書式 IMAGE：〈書式制御文字列〉
機能 PRINT USING文の〈書式制御文字列〉を指定します。行番号のあるプログラムでだけ使えます。

IOCTL$関数 QB
書式 IOCTL$（〔#〕 *filenumber*)
機能 デバイス・ドライバから制御データ文字列を受け取ります。

IOCTLステートメント QB
書式 IOCTL 〔#〕 *filenumber*, *string*
機能 デバイス・ドライバへ制御データ文字列を送ります。

IP関数 TB
書式 IP（〈数値式〉）
機能 〈数値式〉の値のうち，小数点以下を取った整数値を返します。

JIS関数 TB
書式 JIS（〈整数式〉）
機能 〈整数式〉で表されるシフトJISコードを，対応するJISコードに変換します。

JIS$関数 QB
書式 JIS$(*stringexpression*)
機能 文字列の先頭の1文字をJISコードに変換します。

JIS$ (DISKモード) N88
書式 JIS$（〈文字列〉）
機能 2バイト系日本語文字の漢字コードを得ます。

KACNV$ (DISKモード) N88
書式 KACNV$（〈文字列〉）
機能 2バイト系全角文字を，対応する1バイト系の英数カナ文字に変換します。

KANJI$関数 TB
書式 KANJI$（〈文字列式〉）
機能 〈文字列式〉の中に1バイト系半角文字があると，それを2バイト系全角文字に変換して返します。

KCHR$関数 TB
書式 KCHR$（〈整数式〉）
機能 〈整数式〉で表されるシフトJISコードを，対応する漢字，かな文字，英数字，記号のどれかに変換して返します。

KCPOS関数 TB
書式 KCPOS（〈文字列式1〉,〈文字列式2〉）
KCPOS（〈文字列式1〉,〈文字列式2〉〈整数式〉）
機能 〈文字列式2〉内のいずれかの文字が，〈文字列式1〉で最初に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，右へ進みます。全角文字も半角文字も1文字として検索します。

KCPOSR関数 TB
書式 KCPOSR（〈文字列式1〉,〈文字列式2〉）
KCPOSR（〈文字列式1〉,〈文字列式2〉,〈整数式〉）
機能 〈文字列式2〉内のいずれかの文字が，〈文字列式1〉で最後に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，左へ進みます。全角文字も半角文字も1文字として検索します。

KEEP TB
使用例 KEEP
KEEP Mysub
機能 ［引数なし］マークが付いている行だけを抽出します。
［引数一つ］指定した行のまとまりだけを抽出します。

KEXT$関数 QB
書式 KEXT$(*stringexpression*, *func*)
機能 文字列から1バイト文字，または2バイト文字だけを抽出します。

KEXT$ (DISKモード) N88
書式 KEXT$（〈文字列〉,〈機能〉）
機能 文字列の中から1バイト系英数カナ文字だけ，あるいは2バイト系日本語文字だけのどちらかを抜き出します。

KEY TB
使用例 KEY
KEY to Mykeys
KEY from Mykeys
機能 ［引数なし］よく使うキー操作をあるキーに定義します。
［引数一つ］定義したキー操作をファイルに保存したり，呼び出したりします。

KEY N88
書式 KEY〈キー番号〉,〈文字列〉
機能 キーボードの上部にあるファンクション・キーに文字列を定義します。

KEYステートメント QB
書式 KEY *n*, *stringexpression*
KEY LIST

KEY ON
KEY OFF

機能 ファンクション・キーにソフトキー文字を割り当て，それを画面最下行に表示します。またソフトキー文字列の表示をオン／オフします。

KEY(n)ステートメント　QB

書式 KEY(n) ON
KEY(n) OFF
KEY(n) STOP

機能 指定したキーのトラッピングをオン／オフします。

KEY LIST　N88

書式 KEY LIST

機能 ファンクション・キーの内容を画面に表示します。

KEY ON／OFF／STOP　N88

書式 1) KEY〔(〈キー番号〉)〕ON
2) KEY〔(〈キー番号〉)〕OFF
3) KEY〔(〈キー番号〉)〕STOP

機能 ファンクション・キーによる割り込みの許可，禁止，停止を定義します。

KILLステートメント　QB

書式 KILL *filespec*

機能 指定したファイルをディスクから削除します。

KILL (DISKモード)　N88

書式 KILL 〈ファイル・ディスクリプタ〉

機能 ディスク上のファイルを削除します。

KINPUT　N88

書式 KINPUT 〈変数名〉

機能 キーボードから入力された2バイト系日本語文字を，文字変数に代入します。

KINSTR関数　QB

書式 KINSTR([*start*,] *stringexpression1*, *stringexpression2*

機能 ある文字列の中で別の文字列を検索し，それが最初に見つかった文字位置を返します。

KINSTR (DISKモード)　N88

書式 KINSTR (〔〈位置〉,〕〈文字列1〉,〈文字列2〉)

機能 2バイト系日本語文字を含む文字列の中から指定文字列を捜して，その文字の位置を得ます。

KLEN関数　N88/TB

書式 KLEN (〈文字列式〉)

機能 漢字，かな文字を含む文字列の文字数を返します。

KLEN (DISKモード)　N88

書式 KLEN (〈文字列〉〔,〈機能〉〕)

機能 2バイト系日本語文字を含む文字列中の，特定タイプの文字の合計数を得ます。

KMID$ (DISKモード)　N88

書式 KMID$ (〈文字列〉,〈式1〉〔,〈式2〉〕)

機能 2バイト系日本語文字を含む文字列の中から，任意の長さの文字列を抜き出します。

KMID$関数　QB

書式 KMID$ (*stringexpression*, *start* [, *count*])

機能 指定した文字列の一部を取り出します。

KMID$ステートメント　QB

書式 KMID$(*stringvariable*, *start* [, *count*]) = *stringexpression*

機能 指定文字列の一部を別の文字列で置き換えます。

KNCPOS関数　TB

書式 KNCPOS (〈文字列式1〉,〈文字列式2〉)
KNCPOS (〈文字列式1〉,〈文字列式2〉,〈整数式〉)

機能 〈文字列式2〉内にない文字が，〈文字列式1〉で最初に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，右へ進みます。全角文字も半角文字も1文字として検索します。

KNCPOSR関数　TB

書式 KNCPOSR (〈文字列式1〉,〈文字列式2〉)
KNCPOSR (〈文字列式1〉,〈文字列式2〉,〈整数式〉)

機能 〈文字列式2〉内にない文字が〈文字列式1〉で最後に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索が始まり，左へ進みます。全角文字も半角文字も1文字として検索します。

KNJ$ (DISKモード)　N88

書式 KNJ$ (〈文字列〉)

機能 漢字コード文字列4桁を2バイト系日本語文字1文字に変換します。

KORD関数　TB

書式 KORD (〈文字列式〉)

機能 〈文字列式〉の中の先頭の文字のコードを返します。

KPLOAD (DISKモード)　N88

書式 KPLOAD 〈漢字コード〉,〈整数型配列名〉

機能 利用者定義文字パターンをシステムに登録します。

KPOS関数　QB

書式 KPOS(*stringexpression*, *characternumber*)

機能 文字列中の指定した文字位置までのバイト数を返します。

KPOS関数　TB

書式 KPOS (〈文字列式1〉,〈文字列式2〉)
KPOS (〈文字列式1〉,〈文字列式2〉,〈整数式〉)

機能 〈文字列式2〉が，〈文字列式1〉内で最初に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索が始まり，右へ進みます。全角文字も半角文字も1文字として検索します。

KPOSR関数　TB

書式 KPOSR (〈文字列式1〉,〈文字列式2〉)
KPOSR (〈文字列式1〉,〈文字列式2〉,〈整数式〉)

機能 〈文字列式2〉が，〈文字列式2〉内で最後に現れる位置を返します。3番目の引数として〈整数式〉がある場合には，〈文字列式1〉のうち，その数値にあたる文字位置から検索がはじまり，左へ進みます。全角文字も半角文字も1文字として検索します。

KTN$関数　QB

書式 KTN$(*stringexpression*)

機能 文字列の先頭の1文字を句点コード，またはASCII文字コードに変換します。

KTYPE関数　TB

書式 KTYPE (〈文字列式〉)

機能 〈文字列式〉の，先頭の文字タイプを返します。1バイト系英数カナ文字の場合には0，2バイト系全角文字の場合には1,2バイト系半角文字の場合には2,これ以外の文字の場合には−1になります。

KTYPE (DISKモード)　N88

書式 KTYPE (〈文字列〉,〈式〉)

機能 2バイト系日本語文字を含む文字列中の，指定位置の文字のタイプを得ます。

LANGUAGE　TB

使用例 LANGUAGE ENGLISH

LANGUAGE JAPANESE

機能 ［引数一つ］メニュー・バー，ファンクション・キー，メッセージの表示を日本語または英語に切り換えます。

LBOUND関数 TB

書式 LBOUND（〈配列引数〉，〈整数式〉）
LBOUND（〈配列引数〉）

機能 引数が２個あるときは，〈配列引数〉で表された配列について，〈整数式〉番目の次元の添字の最小値（下限）を返します。２番目の引数がないときは，〈配列引数〉で表された配列の１番目の添字の最小値（下限）を返します。その配列は１次元の配列でなければなりません。

LBOUND関数 QB

書式 LBOUND(*array* [, *dimension*])

機能 配列の指定した次元で使うことができる，添字の下限（最小値）を返します。

LCASE$関数 QB

書式 LCASE$(*stringexpression*)

機能 すべての文字を小文字に変換した文字列を返します。

LEFT$関数 QB

書式 LEFT$(*stringexpression, n*)

機能 指定した文字列の左端からｎ個の文字を取り出します。

LCASE$関数 TB

書式 LCASE$（〈文字列式〉）

機能 〈文字列式〉で表される文字列の中にある英字の大文字をすべて小文字に変換して返します。

LEFT$ N88

書式 LEFT$（〈文字列〉，〈式〉）

機能 文字列の左側から任意の長さの文字列を抜き出します。

LEN N88/QB

書式 LEN（〈文字列〉）

機能 文字列の合計バイト数を得ます。

LEN関数 TB

書式 LEN（〈文字列式〉）

機能 〈文字列式〉で表される文字列の長さ（文字の数）を返します。

LET文 N88/TB/QB

書式 LET〈数値変数〉．．．．〈数値変数〉＝〈数値式〉
LET〈文字列変数式〉．．．．〈文字列変数式〉＝〈文字列式〉
〈文字列変数式〉：：〈文字列変数〉
〈文字列変数〉〈部分文字列式〉

機能 右辺の式を計算して，その結果を左辺にある変数に代入します。

LIBRARY文 TB

書式 LIBARY〈ダブル・クォーテーション・マークで囲まれた文字列〉．．．．
〈ダブル・クォーテーション・マークで囲まれた文字列〉

機能 使用するライブラリ・ファイルを指定します。

LINE N88/QB

書式 LINE〔｛(Wx1, Wy1) ｜ STEP(x1, y1)｝〕−｛(Wx2, Wy2) ｜ STEP(x2, y2)｝〔，〈パレット番号１〉〕
〔，｛B ｜ BF｝〕〔，｛〈ライン・スタイル〉｜〈パレット番号２〉｜〈タイル・ストリング〉｝〕

機能 指定した２点間に直線を描きます（パラメータ指定により４角形も描きます）。

LINE INPUT N88/QB

書式 LINE INPUT 〔〈プロンプト文〉；〕〈文字変数名〉

機能 キーボードから入力されるデータ（255バイト以内）を，区切ることなく一括して文字変数に代入します。

LINE INPUT文 TB

書式 LINE INPUT〈文字列リスト〉
LINE INPUT〈入力オプション〉．．．．〈入力オプション〉：〈文字列変数リスト〉
LINE INPUT #〈整数式〉：〈文字列変数リスト〉
LINE INPUT #〈整数式〉，〈ファイル入力オプション〉．．．．〈ファイル入力オプション〉：〈文字列変数リスト〉
〈文字列変数リスト〉：：〈文字列変数式〉．．．．〈文字列変数式〉
〈文字列変数式〉：：〈文字列変数〉
〈文字列変数〉〈部分文字列式〉
〈ファイル入力オプション〉：：〈入力オプション〉
IN MISSING THEN（ジャンプ）
〈入力オプション〉：：PROMPT〈文字列式〉
TIMEOUT〈数値式〉
ELAPSED〈数値変数〉
〈ジャンプ〉：：EXIT DO
EXIT FOR
〈行番号〉

機能 キーボードやファイルから入力した行を〈文字列変数リスト〉の〈文字列変数式〉に代入します。

LINE INPUT # N88/QB

書式 LINE INPUT #〈ファイル番号〉，〈文字型変数名〉

機能 シーケンシャル・ディスク・ファイルより，１行（255バイト以内）単位のデータを一括して文字型変数に読み込みます。

LINE INPUT WAIT N88

書式 LINE INPUT WAIT〈待ち時間〉，〔〈プロンプト文〉｛；｜，｝〕〈文字型変数名〉

機能 キーボードから入力されるデータを変数に代入します。その際，入力待ち時間を制限することができます。

LIST TB

使用例 LIST
LIST 10-30

機能 ［引数なし］カレントプログラムを印刷します。
［引数一つ］指定した行のまとまりを印刷します。

LIST/LLIST N88

書式 1）LIST 〔〈始点行番号〉〕〔−〈終点行番号〉〕
2）LLIST 〔〈始点行番号〉〕〔−〈終点行番号〉〕

機能 メモリ上にあるプログラムの全部，または一部を表示あるいは印刷します。

LOAD N88

書式 LOAD〈ファイル・ディスクリプタ〉〔，R〕

機能 プログラムをメモリにロードします。

LOAD TB

使用例 LOAD Mylibs

機能 ［引数一つ］ライブラリ・ファイルをメモリにロードします。

LOC N88

書式 LOC（〈ファイル番号〉）

機能 ファイル中での論理的な現在位置を得ます。

LOC関数 N88

書式 LOC(*filenumber*)

機能 ファイル内で次に入出力を行なう位置を返します。

LOCAL文 TB

書式 LOCAL〈ローカル項目〉．．．．〈ローカル項目〉
〈ローカル項目〉：：〈単一数値変数〉
〈単一文字列変数〉
〈配列〉〈添字範囲〉

機能　ローカル変数を宣言します。

LOCATE　N88

書式　LOCATE 〔〈X〉〕〔,〈Y〉〕〔,〈カーソル・スイッチ〉〕

機能　テキスト画面のカーソルを指定位置へ移動します。

LOCATEステートメント　QB

書式　LOCATE [*row*] [, [*column*] [, [*cursor*] [, [*start*, *stop*]]]]

機能　カーソルを指定した位置に移動します。

LOCATE　TB

書式　LOCATE a$

機能　[引数一つ] 指定した文字列を含む行のリストを表示します。

LOCK...UNLOCKステートメント　QB

書式　LOCK [#] *filenumber* [, {*record* | [*start*] TO end }]

.

.

.

UNLOCK [#] *filenumber* [, {*record* | [*start*] TO end }]

機能　オープンしているファイルの全部または一部に対する，他のプロセスからのアクセスを制御します。

LOF　N88

書式　LOF (〈ファイル番号〉)

機能　ファイルの大きさを得ます。

LOF関数　QB

書式　LOF(*filenumber*)

機能　指定したファイルの長さをバイト単位で返します。

LOG関数　N88/TB/QB

書式　LOG (〈数値式〉)

機能　〈数値式〉の値の自然対数を返します。

LOG10関数　TB

書式　LOG10 (〈数値式〉)

機能　〈数値式〉の値の常用対数を返します。

LOG2関数　TB

書式　LOG2 (〈数値式〉)

機能　〈数値式〉の値の，底を2とした対数を返します。

LPOS　N88

書式　LPOS (〈式〉)

機能　現在のプリンタのヘッド位置を得ます。

LTRIM$関数　TB/QB

書式　LTRIM$ (〈文字列式〉)

機能　〈文字列式〉で表される文字列の先頭に空白がある場合に，その空白を取り除いて返します。

LPRINT　N88/QB

書式　LPRINT 〔〈式〉〔{ , | ; }〈式〉…〕〔{ , | ; }〕〕

機能　プリンタにデータを出力します。

LPRINT USING　N88/QB

書式　LPRINT USING〈書式制御文字列〉;〈式〉〔{ , | ; }〈式〉…〕

〔{ , | ; }〕

機能　文字列，数値などのデータを編集し，プリンタに出力します。

LSET/RSET (DISKモード)　N88

書式　1) LSET 〈文字変数〉=〈文字列〉

2) RSET 〈文字変数〉=〈文字列〉

機能　ランダム・ファイル・バッファのフィールドにデータを代入します。

LSETステートメント　QB

書式　LSET*stringvariable*=*stringexpression*

機能　PUTステートメントでファイルに書き込むデータを，メモリからランダム・アクセス・ファイルのバッファに移します。このステートメントは，レコード変数を別のレコード変数にコピーしたり，文字列を左詰めにして文字列変数に格納します。

MAP　N88

書式　MAP (〈数式〉,〈機能〉)

機能　スクリーン座標，ワールド座標の相互変換を行ないます。

MARK　TB

使用例　MARK

MARK Myfunc

機能　[引数なし] カレント・プログラム全体にマークを付けます。

[引数一つ] 指定した行のまとまりにマークを付けます。

MAT代入文　TB

書式　MAT〈代入文〉

〈代入文〉::〈数値配列〉=〈数値配列式〉

〈文字列配列〉=〈文字列配列式〉

〈文字列配列〉〈部分文字列式〉=〈文字列配列式〉

〈数値配列式〉::〈数値配列〉

〈数値配列〉〈数値配列演算子〉〈数値配列〉

〈数値配列定数〉

〈数値配列定数〉〈添字変更式〉

〈一次子〉

〈一次子〉 * 〈数値配列〉

〈一次子〉 * 〈数値配列定数〉

〈一次子〉 * 〈数値配列〉〈添字変更式〉

〈数値配列関数〉〈数値配列〉

〈数値配列演算子〉::＋または－または＊

〈数値配列定数〉::CONまたはIDNまたはZER

CONまたはIDNまたはZER〈添字変更式〉

〈数値配列関数〉::INVまたはTRN

〈文字列配列式〉::〈文字列配列一次子〉

〈文字列配列一次子〉&〈文字列配列一次子〉

〈文字列要素〉 & 〈文字列配列一次子〉

〈文字列配列一次子〉 & 〈文字列要素〉

〈文字列要素〉

〈文字列配列定数〉

〈文字列要素〉 & 〈文字列配列要素〉

〈文字列配列一次子〉::〈文字列配列〉

〈文字列配列〉 & 〈部分文字列式〉

〈文字列配列定数〉::NUL$

NUL$〈添字変更式〉

機能　右辺の配列式を(数値配列式でも文字列配列式でも)計算し，その結果を左辺の配列に代入します。

MAT INPUT文　TB

書式　MAT INPUT〈mat入力リスト〉

MAT INPUT〈入力オプション〉....〈入力オプション〉:〈mat入力リスト〉

MAT INPUT #〈整数式〉:〈mat入力リスト〉

MAT INPUT #〈整数式〉,〈ファイル入力オプション〉....〈ファイル入力オプション〉:〈mat入力リスト〉

〈mat入力リスト〉::〈入力配列〉....〈入力配列〉

〈入力配列〉::〈配列〉

〈配列〉〈添字変更式〉

〈配列〉(?)

〈ファイル入力オプション〉::〈入力オプション〉

IF MISSING THEN〈ジャンプ〉

〈入力オプション〉::PROMPT〈文字列式〉

TIMEOUT〈数値式〉

ELAPSED〈数値変数〉

〈ジャンプ〉::EXIT DO

EXIT FOR

〈行番号〉

〈応答入力〉::〈入力項目〉....〈入力項目〉

〈入力項目〉.... 〈入力項目〉,
〈入力項目〉::〈ダブル・クォーテーション・マークで囲まれている文字列〉
〈ダブル・クォーテーション・マークで囲まれていない文字列〉

機 能 キーボードやファイルから入力した〈応答入力〉を，〈配列〉の要素に順に代入します。

MAT LINE INPUT文 TB

書 式 MAT LINE INPUT〈行入力リスト〉
MAT LINE INPUT〈入力オプション〉....〈入力オプション〉:〈行入力リスト〉
MAT LINE INPUT #〈整数式〉:〈行入力リスト〉
MAT LINE INPUT #〈整数式〉,〈ファイル入力オプション〉....〈ファイル入力オプション〉:〈行入力リスト〉
〈行入力リスト〉::〈次元変更済の文字列配列〉....
〈次元変更済の文字列配列〉
〈次元変更済の文字列配列〉::〈文字列配列〉
〈文字列配列〉〈添字変更式〉
〈ファイル入力オプション〉::〈入力オプション〉
IF MISSION THEN〈ジャンプ〉
〈入力オプション〉PROMPT〈文字列式〉
TIMEOUT〈数値式〉
ELAPSED〈数値変数〉
〈ジャンプ〉::EXIT DO
EXIT FOR
〈行番号〉

機 能 キーボードやファイルから入力した行を〈行入力リスト〉の文字列配列に代入します。

MAT PLOT文 TB

書 式 MAT PLOT POINT:〈matプロット配列〉
MAT PLOT LINES:〈matプロット配列〉
MAT PLOT AREA:〈matプロット配列〉

機 能 2次元の数値配列に格納されている点をプロットします。

MAT PRINT文 TB

書 式 MAT PRINT〈mat出力リスト〉
MAT PRINT〈書式制御オプション〉〈mat書式制御リスト〉
MAT PRINT #〈整数式〉:〈mat出力リスト〉
MAT PRINT #〈整数式〉,〈ファイル出力オプション〉....〈ファイル出力オプション〉:〈mat出力リスト〉
MAT PRINT #〈整数式〉,〈ファイル書式制御オプション〉....〈ファイル書式制御オプション〉:〈mat書式制御リスト〉
〈mat出力リスト〉::〈配列〉...〈区切り記号〉〈配列〉
〈配列〉...〈区切り記号〉〈配列〉〈区切り記号〉
〈mat書式制御リスト〉::〈配列〉...〈配列〉
〈配列〉...〈配列〉;
〈区切り記号〉::，または;
〈書式制御オプション〉::USING〈文字列式〉
USING〈行番号〉
〈ファイル書式制御オプション〉::〈書式制御オプション〉
〈ファイル出力オプション〉
〈ファイル出力オプション〉::IF THERE THEN〈ジャンプ〉
〈ジャンプ〉::EXIT DO
EXIT FOR
〈行番号〉

機 能 配列の要素を画面やファイルに出力します。

MAT READ文 TB

書 式 MAT READ〈読み込み配列リスト〉
MAT READ IF MISSION THEN〈ジャンプ〉:〈読み込み配列リスト〉
MAT READ #〈整数式〉:〈読み込み配列リスト〉
MAT READ #〈整数式〉,〈読み込みオプション〉....〈読み込みオプション〉:〈読み込み配列リスト〉
〈読み込み配列リスト〉::〈読み込み配列〉....〈読み込み配列〉
〈読み込み配列〉::〈配列〉
〈配列〉〈添字変更式〉
〈読み込みオプション〉::〈レコード移動〉
IF MISSING THEN〈ジャンプ〉
BYTES〈整数式〉
〈レコード移動〉::BIGIN
END
NEXT
SAME
〈ジャンプ〉::EXIT DO
EXIT FOR
〈行番号〉

機 能 〈読み込み配列リスト〉のそれぞれの配列の要素に値を代入します。

MAT REDIM文 TB

書 式 MAT REDIM〈配列変数〉〈添字変更式〉....〈配列変数〉〈添字変更式〉

機 能 〈添字変更式〉の設定に従って，配列の次元を変更します。

MAT WRITE文 TB

書 式 MAT WRITE #〈整数式〉:〈mat書き出しリスト〉
MAT WRITE #〈整数式〉,〈書き出しオプション〉....〈書き出しオプション〉:〈mat書き出しリスト〉
〈mat書き出しリスト〉::〈配列〉....〈配列〉
〈配列〉::〈数値配列〉
〈文字列配列〉
〈書き出しオプション〉::〈レコード移動〉
IF THERE THEN〈ジャンプ〉
〈レコード移動〉::BEGIN
END
NEXT
SAME
〈ジャンプ〉::EXIT DO
EXIT FOR
〈行番号〉

機 能 〈mat書き出しリスト〉の配列の要素を，指定されたファイル順に書き出します。

MAX関数 TB

書 式 MAX (〈数値式1〉,〈数値式2〉)

機 能 〈数値式1〉の値と〈数値式2〉の値のうち，大きいほうを返します。

MAXLEN関数 TB

書 式 MAXLEN (〈文字列変数〉)

機 能 〈文字列変数〉の最大長（最大の文字数）を返します。

MAXNUM関数 TB

書 式 MAXNUM

機 能 コンピュータで計算できる最大の数を返します。引数はありません。

MAXSIZE関数 TB

書 式 MAXSIZE (〈配列引数〉)

機 能 配列引数の配列宣言によって許される全要素の個数を返します。True BASICでは2^31を返します。

MENUS TB

書 式 MENUS

機 能 [引数なし] メニュー・バー，スクロール・バー，マウスの表示を消します。

MERGE (DISKモード) N88

書 式 MERGE〈ファイル・ディスクリプタ〉

機 能 メモリ上のプログラムに，ディスク上のプログラム・ファイルを合成します。

MID$ N88/QB

書 式 MID$ (〈文字変数〉,〈式1〉〔,〈式2〉〕)=〈文字列〉

機 能 文字列の一部を置き換えます。

MID$ N88/QB

書式 MID$ (〈文字列〉, 〈式1〉[, 〈式2〉])
機能 文字列の中から任意の長さの文字列を抜き出します。

MIN関数 TB

書式 MIN 〈数値式1〉, 〈数値式2〉
機能 〈数値式1〉の値と〈数値式2〉の値のうち、小さいほうを返します。

MKD$, MKI$, MKL$, MKS$関数 QB

書式 MKI $(*integerexpression*)
MKS$(*single-precision-expression*)
MKL$(*long-integer-expression*)
MKI$(*double-precision-expression*)
機能 数値を文字列に変換します。

MKDIRステートメント QB

書式 MKDIR*pathname*
機能 新しいディレクトリを作ります。

MKI$/MKS$/MKD$ N88

書式 MKI$ (〈整数値〉)
MKS$ (〈単精度実数値〉)
MKD$ (〈倍精度実数値〉)
機能 数値データをその数値の内部表現に対応した文字列に変換します。

MKSMBF$, MKDMBF$関数 QB

書式 MKSMBF$(*single-precision-expression*)
MKDMBF$(*double-precision-expression*)
機能 IEEE形式の数値を, Microsoftバイナリ形式の数値を含む文字列に変換します。

MOD関数 TB

書式 MOD (〈数値式1〉, 〈数値式2〉)
機能 〈数値式1〉の値を, 〈数値式2〉の値でわり, 〈数値式2〉と同じ符号の余りを返します。

MODE TB

使用例 MODE
機能 [引数なし] Shellサブルーチンによって崩れた画面を回復します。

MODULE構文 TB

書式 MODULE構文::MODULE 〈識別子〉
... 〈モジュール・ヘッダ〉
... 〈プロシージャ部〉
END MODULE
〈モジュール・ヘッダ〉::... 〈モジュール文〉
〈モジュール文〉::〈public文〉
〈share文〉
〈private文〉
〈その他の文〉
〈プロシージャ部〉::... 〈プロシージャ〉
機能 モジュールを定義します。

MON (DISKモード) N88

書式 MON
機能 モニタモードに入ります。

MOVE TB

使用例 MOVE
MOVE Mysub.Myfunc
機能 [引数なし] マークが付いている行をカーソルのある行以降に移動します。
[引数二つ] 行のまとまりを, 指定した行に移動します。

NAME (DISKモード) N88

書式 NAME 〈旧ファイル・ディスクリプタ〉 AS 〈新ファイル・ディスクリプタ〉
機能 ディスク・ファイルの名前を変更します。

NAMEステートメント QB

書式 NAME*oldfilename* AS *newfilename*
機能 ディスク・ファイルやディレクトリの名前を変えます。

NCPOS関数 TB

書式 NCPOS (〈文字列式1〉, 〈文字列式2〉)
NCPOS (〈文字列式1〉, 〈文字列式2〉, 〈整数式〉)
機能 〈文字列式2〉内にない文字が, 〈文字列式1〉で最初に現れる位置を返します。3番目の引数として〈整数式〉がある場合には, 〈文字列式1〉のうち, その数値にあたる文字位置から検索がはじまり, 右へ進みます。

NCPOSR関数 TB

書式 NCPOSR (〈文字列式1〉, 〈文字列式2〉)
NCPOSR (〈文字列式1〉, 〈文字列式2〉, 〈整数式〉)
機能 〈文字列式2〉内にない文字が〈文字列式1〉で最後に現れる位置を返します。3番目の引数として〈整数式〉がある場合には, 〈文字列式1〉のうち, その数値にあたる文字位置から検索が始まり, 左へ進みます。

NEW N88

書式 NEW
機能 メモリ上にあるプログラムを抹消し, すべての変数を初期化します。

NEW TB

使用例 NEW
NEW Myfile
機能 [引数なし] エデット・ウィンドウのプログラムを消去します。
[引数一つ] エデット・ウィンドウのプログラムを消去し, 新しいプログラム名を指定します。

NEW ON N88

書式 NEW ON 〈式〉
機能 システムを再起動します。

NOLET TB

書式 NOLET
機能 [引数なし] プログラムの中のLETを不要にします。

NUL$配列定数 TB

書式 NUL$ 〈添字変更式〉
NUL$
機能 要素がすべてヌルストリングスからなっている文字列配列を返します。NUL が使えるのは, MAT代入文内だけです。〈添字変更式〉があるときは, その〈添字変更式〉で指定した次元の配列が生成されます。〈添字変更式〉がなければ, MAT代入文で代入された配列の次元と一致します。

NUM関数 TB

書式 NUM 〈文字列式〉
機能 〈文字列式〉で表される文字列をIEEEの8バイト書式として扱い, 数値に変換して返します。

NUM$関数 TB

書式 NUM$関数 〈数値式〉
機能 〈数値式〉の値を, IEEEの8バイト書式を使って8桁の文字列に変換して返します。

OCT$ N88/QB

書式 OCT$ 〈数式〉
機能 10進数を8進数に変換し, その文字列を得ます。

OLD TB

使用例 OLD HANOI
機能 [引数一つ] ディスクに保存されているファイルを呼び出し

QB＝Quick BASIC TB＝True BASIC

てオープンします。

ON COM GOSUB　N88

書式　ON COM〔(〈回線番号〉)〕GOSUB〈行番号〉
機能　RS-232C回線からの割り込みが発生したとき，分岐する処理ルーチンの開始行を指定します。

ON ERROR GOTO　N88/QB

書式　ON ERROR GOTO〈行番号〉
機能　エラーが起こったときに分岐する処理ルーチンの開始行を定義します。

ON eventステートメント　QB

書式　ON *event* GOSUB {*linenumber*｜*linelabel*}
機能　イベント・ラッピング・ルーチンの最初の行を指定します。

ON…GOSUB／ON…GOTO　N88

書式　1) ON〈式〉GOSUB〈行番号〉〔,〈行番号〉…〕
　2) ON〈式〉GOTO〈行番号〉〔,〈行番号〉…〕
機能　指定されたいずれかの行に分岐します。

ON…GOSUB／ON…GOTOステートメント　QB

書式　ON*expression*GOSUB {*line-number-lost*｜*line-label-list*}
　ON*expression*GOTO {*line-number-lost*｜*line-label-list*}
機能　式の値に応じて，指定した行番号／行ラベルリストの一つに分岐します。

ON GOSUB文　TB

書式　ON〈整数式〉GOSUB〈行番号リスト〉
　ON〈整数式〉GOSUB〈行番号リスト〉ELSE〈単一文〉
　〈行番号リスト〉::〈行番号〉....〈行番号〉
機能　〈整数式〉の計算値に従って，〈行番号〉にジャンプするか，〈単一文〉を実行します。対応するRETURN文が実行されるとON GOSUB文の次の文に戻ります。

ON GOTO文　TB

書式　ON〈整数式〉GOTO〈行番号リスト〉
　ON〈整数式〉GOTO〈行番号リスト〉ELSE〈単一文〉
　〈行番号リスト〉::〈行番号〉....〈行番号〉
機能　〈整数式〉の計算値に従って，〈行番号〉にジャンプするか，〈単一文〉を実行します。

ON HELP GOSUB　N88

書式　ON HELP GOSUB〈行番号〉
機能　[HELP] キーによる割り込み処理ルーチンの開始行を定義します。

ON KEY GOSUB　N88

書式　ON KEY GOSUB〈行番号〉〔,〈行番号〉…〕
機能　ファンクション・キーによる割り込みルーチンの開始行を定義します。

ON PEN GOSUB　N88

書式　ON PEN GOSUB〈行番号〉
機能　ライトペンが押されたときの割り込み処理ルーチンの開始行を定義します。

ON PLAY GOSUB　N88

書式　ON PLAY (〈チャンネル番号〉,〈残りバイト数〉GOSUB〈行番号〉
機能　PLAY割り込み処理ルーチンの開始行を定義します。

ON STOP GOSUB　N88

書式　ON STOP GOSUB〈行番号〉
機能　[STOP] キーによる割り込み処理ルーチンの開始行を定義します。

ON TIME$ GOSUB　N88

書式　ON TIME$="〈時刻〉"GOSUB〈行番号〉
機能　内蔵クロックによる割り込みの発生時刻と，そのとき分岐する処理ルーチンの開始行を定義します。

ON UEVENT GOSUBステートメント　QB

書式　ON UEVENT GOSUB {*linenumber*｜*linelabel*}
機能　ユーザー定義のイベントをトラッピングするサブルーチンを定義します。

OPEN文　TB

書式　OPEN #〈整数式〉: NAME〈文字列式〉
　OPEN #〈整数式〉: NAME〈文字列式〉,〈オープン・リスト〉
　OPEN #〈整数式〉: PRINTER
　OPEN #〈整数式〉: SCREEN〈スクリーン座標〉
　〈オープンリスト〉::〈オープン句〉....〈オープン句〉
　〈スクリーン座標〉::〈数値式1〉,〈数値式2〉,〈数値式3〉,〈数値式4〉
　〈オープン句〉:: ACCESS INPUT
　ACCESS OUTPUT
　ACCESS OUTIN
　ACCESS〈文字列式〉
　CREATE NEW
　CREATE OLD
　CREATE NEWOLD
　CREATE〈文字列式〉
　ORGANIZATION TEXT
　ORGANIZATION STREAM
　ORGANIZATION RANDOM
　ORGANIZATION RECORD
　ORGANIZATION BYTE
　ORGANIZATION〈文字列式〉
　RECSIZE〈整数式〉
機能　ファイルまたはプリンタ，ウィンドウをオープンします。

OPEN　N88

書式　OPEN〈ファイル・ディスクリプタ〉〔FOR〈モード〉〕AS〔#〕〈ファイル番号〉
機能　ファイルを開きます。

OPENステートメント　QB

書式　1 OPEN *file* [FOR *mode*1] [ACCESS *access*] [*lock*] AS [#] *filenum* [LEN=*reclen*]
　2 OPEN *mode*2, [#] *filenum*, *file* [, *reclen*]
機能　ファイルやデバイスへの入出力を可能にします。

OPEN COMステートメント　QB

書式　OPEN "COM *n*:*optist1 optilist2*" [FOR*mode*] AS [#] *filenum* [LEN=*reclen*]
機能　入出力を行なうため通信チャンネルをオープンして初期化します。

OPTION ANGEL文　TB

書式　OPTION ANGEL DEGREES
　OPTION ANGEL RADIANS
機能　三角関数やグラフィック変換で使用する角度を，度数またはラジアンに切り換えます。

OPTION ARITHMETIC文　TB

書式　OPTION ARITHMETIC NATIVE
　OPTION ARITHMETIC STANDARD
機能　現在のTrue BASICのバージョンでは無効ですが，ANSI規格との互換性を保つためにあります。

OPTION BASE文　TB

書式　OPTION BASE〈符号つきの整数〉
機能　下限が設定されていない配列や配列定数の下限を指定します。

OPTION BASE　N88/QB

書式　OPTION BASE |0|1|
機能　配列の添字の最小値を指定します。

OPTION COLLATE　TB

書式　OPTION COLLATE NATIVE
OPTION COLLATE STANDARD

機能　現在のTrue BASICのバージョンでは無効ですが，ANSI規格との互換性を保つためにあります。

OPTION NOLET文　TB

書式　OPTION NOLET

機能　LETなしでLET文が書けるようになります。

OPTION TYPO文　TB

書式　OPTION TYPO

機能　配列以外の変数が，宣言しないと使用できなくなります。

ORD関数　TB

書式　ORD（〈文字列式〉）

機能　〈文字列式〉で表される文字列のASCIIコードを返します。

OUT　N88/QB

書式　OUT 〈ポート番号〉，〈式〉

機能　出力ポートに１バイトのデータを送出します。

PACKBサブルーチン　TB

書式　CALL PACKB 〈文字列式〉，〈整数式1〉，〈整数式2〉，〈整数式3〉

機能　整数をある文字列内のビットにパック（圧縮）するルーチンです。UNPACKB関数で，パックされた文字列を，整数に復元（アンパック）します。

(1) PAINT　N88

書式　PAINT ｜(Wx，Wy)｜〔，〈領域色〉〕〔，〈境界色〉〕
　　　　　｜STEP(x，y)｜

機能　指定された境界色で囲まれた領域を，指定された色でぬります。

(2) PAINT　N88

書式　PAINT｜(Wx，Wy)｜〔，〈タイル・ストリング〉〕〔，〈境界色〉〕
　　　　　｜STEP(x，y)｜

機能　指定された境界色で囲まれた領域を，指定されたタイル・パターンで埋めます。

PAINT ステートメント　QB

書式　PAINT [STEP] (*x*, *y*) [, [*paint*] [, [*bordercolor*] [, *background*]]]

機能　指定した色や模様でグラフィックスの領域を塗りつぶします。

PALETTE/PALETTE USINGステートメント　QB

書式　PALETTE [*attribute*, *color*]
PALETTE USING *arrayname* [(*arrayindex*)]

機能　パレットの色を変えます。

PAUSE文　TB

書式　PAUSE 〈数値式〉

機能　〈数値式〉の秒数だけプログラムの実行を停止します。

PCOPYステートメント　QB

書式　PCOPY *sourcepage*, *destinationpage*

機能　あるスクリーン・ページの内容を別のページにコピーします。

PEEK　N88/QB

書式　PEEK｜〈アドレス〉

機能　メモリ上の指定番地の内容を読み出します。

PEEK関数　TB

書式　PEEK（〈整数式〉）

機能　メモリ上の指定されたバイトの内容を照会します。〈整数式〉の値で指定されたメモリ上の番地の内容を，0から255の数値で返します。

PEN　N88

書式　PEN（〈機能〉）

機能　ライトペンから情報を得ます。

PEN ON/OFF/STOP　N88

書式　1) PEN ON
2) PEN OFF
3) PEN STOP

機能　ライトペンによる割り込みの許可，禁止，停止を制御します。

PI関数　TB

書式　PI

機能　円周率を返します。約3.14159265...になります。引数はありません。

PICTURE構文　TB

書式　〈picture構文〉：：〈picture文〉
　　　　　....
　　　　　END PICTURE
〈picture文〉：：PICTURE 〈識別子〉
　　　　　PICTURE 〈識別子〉〈サブルーチン・パラメータ・リスト〉
〈サブルーチン・パラメータ・リスト〉：：〈サブルーチン・パラメータ〉....〈サブルーチン・パラメータ〉
〈サブルーチン・パラメータ〉：：〈単一数値変数〉

PLAY文　TB

書式　PLAY 〈文字列式〉

機能　〈文字列式〉で曲を定義します。

PLAY　N88

書式　PLAY〔#〈モード番号〉，〕〔〈文字列1〉〕〔，〈文字列2〉〕〔，〈文字列3〉〕〔，〈文字列4〉〕〔，〈文字列5〉〕〔，〈文字列6〉〕

機能　音楽演奏を行ないます

PLAY関数　QB

書式　PLAY(*n*)

機能　BGMキューに含まれる音符の数を返します。

PLAYステートメント　QB

書式　PLAY*commandstring*

機能　文字列で指定した音楽を演奏します。

PLAY ALLOC　N88

書式　PLAY ALLOC〔〈CH1バッファ・サイズ〉〕〔，〈CH2バッファ・サイズ〉〕〔，〈CH3バッファ・サイズ〉〕〔，〈CH4バッファ・サイズ〉〕〔，〈CH5バッファ・サイズ〉〕〔，〈CH6バッファ・サイズ〉〕

機能　サウンドバッファの確保および初期化をします。

PLAY ON/OFF/STOP　N88

書式　1) PLAY ON
2) PLAY OFF
3) PLAY STOP

機能　PLAY割り込みの許可，禁止，停止を制御します。

PLAY ON, OFF, STOPステートメント　QB

書式　PLAY ON
PLAY OFF
PLAY STOP

機能　PLAY ONステートメントは，BGMバッファのイベント・ラッピングを可能にします。
PLAY OFFステートメントは，BGMバッファのイベント・ラッピングを中止します。
PLAY STOPステートメントは，BGMバッファのイベント・ラッピングを中断します。

PLOT文　TB

書式　PLOT

機能　前のPLOT文やPLOT LINES文がセミコロンで終わってい

た場合に，次のPLOT文やPLOT LINES文の始点とつながらないようにします。

PLOT AREA文 TB

書 式 PLOT AREA：〈座標リスト〉

〈座標リスト〉：：〈座標〉：〈座標〉

〈座標〉：：〈数値式 1〉，〈数値式 2〉

機 能 〈座標リスト〉で定義した線を描き，その線で囲まれた領域を塗りつぶします。

PLOT LINES文 TB

書 式 PLOT LINES：〈座標リスト〉

PLOT〈座標リスト〉

PLOT LINES：〈座標リスト〉

PLOT〈座標リスト〉

〈座標リスト〉：：〈座標〉：〈座標〉

〈座標〉：：〈数値式 1〉，〈数値式 2〉

機 能 〈座標リスト〉で定義された点を結ぶ線を描きます。

PLOT POINTS文 TB

書 式 PLOT POINTS：〈座標リスト〉

PLOT〈座標リスト〉

〈座標リスト〉：：〈座標〉：〈座標〉

〈座標〉：：〈数値式 1〉，〈数値式 2〉

機 能 〈座標リスト〉の点を描きます。

PLOT TEXT文 TB

書 式 PLOT TEXT.AT〈座標〉：〈文字列式〉

〈座標〉：：〈数値式 1〉，〈数値式 2〉

機 能 AT句で指定した点に，〈文字列式〉を描きます。

PMAP関数 QB

書 式 PMAP(*expression*, *function*)

機 能 指定した論理座標を物理座標に，また指定した物理座標を論理座標に変換します。

POINT関数 QB

書 式 1 POINT(*x*, *y*)

2 POINT(*number*)

機 能 画面上の指定したピクセルの色番号を読み取ったり，ピクセルの座標を返します。

POINT N88

書 式 POINT │(Wx, Wy)│

│STEP(x, y)│

機 能 LP（最終参照点）を変更します。

(1) POINT N88

書 式 POINT（〈機能〉）

機 能 LP（最終参照点）の値を得ます。

(2) POINT N88

書 式 POINT（Sx, Sy）

機 能 スクリーン座標の指定された座標に表示されているドットの色を得ます。

POKE N88/QB

書 式 POKE〈アドレス〉，〈式〉

機 能 メモリ上の指定番地へデータを書き込みます。

POKEサブルーチン TB

書 式 CALL POKE（〈整数式1〉，〈整数式2〉）

機 能 メモリ上の指定されたバイトの内容を変更します。

POS N88/QB

書 式 POS（〈数式〉）

機 能 テキスト画面上の現在のカーソルの桁位置を変更します。

POS関数 TB

書 式 POS（〈文字列式 1〉，〈文字列式 2〉）

POS（〈文字列式 1〉，〈文字列式 2〉，〈整数式〉）

機 能 〈文字列式 2〉が，〈文字列式 1〉内で最初に現れる位置を返します。3 番目の引数として〈整数式〉がある場合には，〈文字列式 1〉のうち，その数値にあたる文字位置から検索がはじまり，右へ進みます。

POSR関数 TB

書 式 POSR（〈文字列式 1〉，〈文字列式 2〉）

POSR（〈文字列式 1〉，〈文字列式 2〉，〈整数式〉）

機 能 〈文字列式 2〉が，〈文字列式 1〉内で最後に現れる位置を返します。3 番目の引数として〈整数式〉がある場合には，〈文字列式 1〉のうち，その数値にあたる文字位置から検索がはじまり，左へ進みます。

PRESET N88

書 式 PRESET │(Wx, Wy)　│〔，〈パレット番号〉〕

│STEP(x, y)│

機 能 画面上の任意の座標のドットを消去します。

PRESETステートメント QB

書 式 PRESET［STEP］(*xcoordinate*, *ycoordinate*)［, *color*］

機 能 画面上に指定した点を描きます。

PRINT文 TB

書 式 PRINT

PRINT〈出力リスト〉

PRINT〈書式制御オプション〉：〈書式制御リスト〉

PRINT#〈整数式〉

PRINT#〈整数式〉：

PRINT#〈整数式〉：〈出力リスト〉

PRINT#〈整数式〉，〈ファイル書式制御オプション〉....

〈ファイル書式制御オプション〉：〈書式制御リスト〉

〈出力リスト〉：：〈出力項目〉...〈区切り記号〉〈出力項目〉

〈出力項目〉...〈区切り記号〉〈出力項目〉〈区切り記号〉

〈書式制御リスト〉：：〈書式制御項目〉....〈書式制御項目〉

〈書式制御項目〉....〈書式制御項目〉：

〈区切り記号〉：：，または；

〈出力項目〉：：〈数値式〉

〈文字列式〉

〈タブ設定〉

ヌル

〈書式制御項目〉：：〈数値式〉

〈文字列式〉

〈タブ設定〉：：TAB（〈整数式〉）

TAB（〈整数式 1〉，〈整数式 2〉）

〈ファイル書式制御オプション〉：：〈書式制御オプション〉

〈ファイル出力オプション〉

〈書式制御オプション〉：：USING〈文字列式〉

USING〈行番号〉

〈ファイル出力オプション〉：：IF THERE THEN〈ジャンプ〉

〈ジャンプ〉：：EXIT DO

EXIT FOR

〈行番号〉

機 能 〈出力項目〉を画面やファイルに出力します。

PRINT N88

書 式 │PRINT│〔〈式〉〕〔│,│〈式〉…〕〔│,│〕

│?　　│　　　　　│;│　　　　　│;│

機 能 画面にデータを出力します。

PRINTステートメント QB

書 式 PRINT［*expressionlist*］［{, │ ; }］

機 能 指定したデータを画面に表示します。

PRINT# N88

書 式 PRINT#〈ファイル番号〉，〔〈式〉〕〔│,│〈式〉…〕〔│,│〕

│;│　　　　　│;│

機 能 ファイルにデータを書き出します。

PRINT USING　N88

書式　PRINT USING 〈書式制御文字列〉;〔｛, ｜ ;｝〈式〉…〕〔｛, ｜ ;｝〕

機能　文字列，数値などのデータを編集し，画面に出力します。

PRINT USINGステートメント　QB

書式　PRINT USING*formatstring; expressionlist* [{, | ;}]

機能　指定した書式を使って文字列や数値を画面に表示します。

PRINT # USING　N88

書式　PRINT # 〈ファイル番号〉, USING 〈書式制御文字列〉;
〈式〉〔｛, ｜ ;｝〈式〉…〕〔｛, ｜ ;｝〕

機能　文字列，数値などのデータを編集し，ファイルに出力します。

PRINT #, PRINT # USINGステートメント　QB

書式　PRINT #*filenumber*, [USING *stringexpression;*] *expres sionlist* [{, | ;}]

機能　データをシーケンシャル・ファイルに書き込みます。

PRIVATE文　TB

書式　PRIVATE 〈プロシージャ名〉 .... 〈プロシージャ名〉
〈プロシージャ名〉::〈識別子〉
〈文字列識別子〉

機能　〈プロシージャ名〉で指定した外部プロシージャ（サブルーチン，関数，ピクチャ）をモジュールの専用に指定します。

PROGRAM文　TB

書式　PROGRAM 〈識別子〉
PROGRAM 〈識別子〉(〈関数定義パラメータ・リスト〉)
〈関数定義パラメータ・リスト〉::〈関数定義パラメータ〉 .... 〈関数定義パラメータ〉
〈関数定義パラメータ〉::〈単一数値変数〉
〈単一文字列変数〉
〈配列パラメータ〉

機能　連結するときなどに引き渡される引数を指定します。

PSET　N88

書式　PSET ｛(Wx, Wy) ｜ STEP(x, y)｝〔,〈パレット番号〉〕

機能　画面上の任意の座標にドットを表示します。

PSETステートメント　QB

書式　PSET [STEP](*xcoordinate, ycoordinate*)[, *color*]

機能　画面上に点を描きます。

PUBLIC文　TB

書式　PUBLIC 〈公用項目〉 .... 〈公用項目〉
〈公用項目〉::〈単一数値変数〉
〈単一文字列変数〉
〈配列〉〈添字範囲〉

機能　変数や配列が，モジュール（またはプログラム・ユニット）の外部からアクセスできるように指定します。

PUT (DISKモード)　N88

書式　PUT〔#〕〈ファイル番号〉〔,〈数式〉〕

機能　ファイル・バッファ中のデータをファイルに書き出します。

PUTステートメント－ファイルI/O　QB

書式　PUT #*filenumber* [, [*recordnumber*] [, *variable*]]

機能　変数やランダム・アクセス・ファイルのバッファの内容をファイルに書き込みます。

PUTステートメント－グラフィックス　QB

書式　PUT [STEP](*x,y*), *arrayname* [(*indices*)] [, *action verb*]

機能　GETステートメントで保存したグラフィックス・イメージを画面上に表示します。

PUT@　N88

書式　1) PUT〔@〕(Sx, Sy),〈配列変数名〉〔(〈添字〉)〕〔,〈条件〉〕〔,〈フォアグラウンド・カラー〉,〈バックグラウンド・カラー〉〕
2) PUT〔 〕(Sx, Sy), KANJI(〈漢字コード〉)〔,〈条件〉〕〔,〈フォアグラウンド・カラー〉,〈バックグラウンド・カラー〉〕

機能　グラフィック・パターンや漢字を画面に表示します。

RAD関数　TB

書式　RAD (〈数値式〉)

機能　度数で与えられた〈数値式〉を，ラジアンに変換します。

RANDOMIZE　N88

書式　RANDOMIZE〔〈式〉〕

機能　新しい乱数系列を設定します。

RANDOMIZEステートメント　QB

書式　RANDOMIZE [*expression*]

機能　乱数ジェネレータを初期化します（乱数系列を再設定します）。

RANDOMIZE文　TB

書式　RANDOMIZE

機能　乱数の新しい種（SEED)を生成します。

READ　N88/QB

書式　READ 〈変数〉〔,〈変数〉…〕

機能　DATAで用意した数値や文字のデータを読み込み，変数に代入します。

READ文　TB

書式　READ 〈変数〉 .... 〈変数〉
READ IF MISSING THEN〈ジャンプ〉:〈変数〉....〈変数〉
READ #〈整数式〉:〈変数〉 .... 〈変数〉
READ #〈整数式〉:〈変数〉 .... 〈変数〉 ,SKIP REST

機能　READ #〈整数式〉,〈読み込みオプション〉 .... 〈読み込みオプション〉:〈変数〉 .... 〈変数〉
READ #〈整数式〉,〈読み込みオプション〉 .... 〈読み込みオプション〉:〈変数〉 .... 〈変数〉 ,SKIP REST
〈読み込みオプション〉::〈レコード移動〉
IF MISSING THEN 〈ジャンプ〉
BYTES 〈整数式〉
〈レコード移動〉::BIGIN
END
NEXT
SAME
〈ジャンプ〉::EXIT DO
EXIT FOR
〈行番号〉
〈変数〉::〈数値変数〉
〈文字列変数〉
〈文字列変数〉〈部分文字列式〉
データリストやファイルのデータを〈変数〉に代入します。

REDIMステートメント　QB

書式　REDIM [SHARED] *variable* (*subscripts*)[AS*type*] [, *variable* (*subscripts*)[AS*type*]] ...

機能　DYNAMIC配列に割り当てた領域を変更します。

RENAME　TB

書式　RENAME Yourfile
RENAME Myfile, Yourfile

機能　[引数一つ] カレント・プログラム名を変更します。
[引数二つ] ディスクに保存されているファイルの名前を変更します。

RENUM　N88

書式　RENUM〔〈新行番号〉〕〔,〈旧行番号〉〕〔,〈増分〉〕

QB＝Quick BASIC　TB＝True BASIC

機能　プログラムの行番号を新しくつけ直します。

REM　N88/TB/QB

書式　REM 〔〈注釈文〉〕
　　　'

機能　プログラムに注釈文を入れます。

REMAINDER関数　TB

書式　REMAINDER（〈数値式1〉, 〈数値式2〉）

機能　〈数値式1〉の値を，〈数値式2〉の値でわり，〈数値式1〉と同じ符号の余りを返します。

REPEAT$関数　TB

書式　REPEAT$（〈文字列式〉, 〈整数式〉）

機能　〈文字列式〉で表される文字列を，〈整数式〉の値の数だけ複写して返します。

REPLACE　TB

使用例　REPLACE
　　　REPLACE CARDS

機能　［引数なし］カレント・プログラムをディスクに上書して保存します。
　　　［引数一つ］カレント・プログラムを，ディスクの指定した名前のファイルに上書して保存します。

RESET文　TB

書式　RESET #〈整数式〉: BEGIN
　　　RESET #〈整数式〉: END
　　　RESET #〈整数式〉: NEXT
　　　RESET #〈整数式〉: SAME
　　　RESET #〈整数式〉: RECORD〈整数式2〉

機能　ファイル・ポインタを，指定した位置に設定します。

RESETステートメント　QB

書式　RESET

機能　すべてのディスク・ファイルをクローズします。

RESTORE文　TB

書式　RESTORE
　　　RESTORE〈行番号〉

機能　データ・ポインタを，データリストの先頭に戻します。

RESTORE　N88

書式　RESTORE〔〈行番号〉〕

機能　READで読むDATA行の先頭行を指定します。

RESTOREステートメント　QB

書式　RESTORE [{*linenumber* | *linelabel*}]

機能　DATAステートメントのデータを，指定した行から読み込みます。

RESUME　N88

書式　1) RESUME　〔0〕
　　　2) RESUME　NEXT
　　　3) RESUME　〈行番号〉

機能　エラー処理ルーチンを終了し，元のプログラムの実行を再開します。

RESUMEステートメント　QB

書式　1 RESUME [0]
　　　2 RESUME NEXT
　　　3 {linenumber | linelabel}

機能　エラー・トラッピング・ルーチンを呼び出した後で，プログラムの実行を継続します。

RETRY文　TB

書式　RETRY

機能　エラーが発生したときに実行していた行に移動します。WHEN構文かHANDLER構文のハンドラ部でだけ使えます。

RETURN　N88/QB

書式　RETURN〔〈行番号〉〕

機能　サブルーチンを終了し，元のプログラムの実行を再開します。

RIGHT$関数　N88/QB

書式　RIGHT$（〈文字列〉, 〈式〉）

機能　文字列の右側から任意の長さの文字列を抜き出します。

RND関数　N88

書式　RND〔〈数式〉〕

機能　乱数を得ます。

RND関数　TB

書式　RND

機能　乱数を返します。引数はありません。実行のたびごとに違った結果が出るようにしたければ，RND関数の前にRANDOMIZE文を使います。

RND関数　QB

書式　RND [(*n*)]

機能　0から1の範囲の，単精度の乱数を返します。

RMDIRステートメント　QB

書式　RMDIR*pathname*

機能　指定したディレクトリを削除します。

ROLL　N88

書式　ROLL〔〈上下方向ドット数〉〕〔,〈左右方向ドット数〉〕〔,{N|Y}〕

機能　グラフィック画面を上下あるいは左右にスクロールさせます。

ROUND関数　TB

書式　ROUND（〈数値式〉, 〈整数式〉）
　　　ROUND（〈数値式〉）

機能　ROUND(x,n)は，xを小数点以下n桁までで四捨五入した値を返します。ROUND(x)は，ROUND(x,0)と同じです。nが負の場合は10の｜n｜乗の桁までで四捨五入した値を返します。

RSETステートメント　QB

書式　RSET*stringvariable*=*stringexpression*

機能　PUTステートメントでファイルに書き込むデータを，メモリからランダム・アクセス・ファイルのバッファに移します。このステートメントは，文字列を右詰めにして文字列変数に格納します。

RTRIM$関数　TB/QB

書式　RTRIM$（〈文字列式〉）

機能　〈文字列式〉で表された文字列の後方に空白がある場合に，その空白を取り除いて返します。

RUN　TB

書式　RUN

機能　［引数なし］カレント・プログラムを実行します。

RUN　N88

書式　1) RUN〔〈行番号〉〕
　　　2) RUN〈ファイル・ディスクリプタ〉〔, R〕

機能　メモリにあるプログラムの実行を開始します。また，ディスクからプログラムをメモリにロードし，そのプログラムを実行します。

RUNステートメント　QB

書式　RUN [{*linenumber* | *linelabel*}]

機能　現在メモリ上にあるプログラムをリスタートしたり，指定したプログラムを実行します。

RUNTIME関数　TB

書式　RUNTIME

機 能 プログラムの実行を開始したときからプロセッサを使用した秒数を返します。引数はありません。RUNTIME関数は，タイムシェアリング・システムでだけ有効です。パソコンでは－1を返します。

SADD関数 QB

書 式 SADD(*stringvariable*)

機 能 指定した文字式のアドレスを返します。

SAVE N88

書 式 SAVE 〈ファイル・ディスクリプタ〉〔, |A / P| 〕

機 能 メモリにあるBASICプログラムをファイルにセーブします。

SAVE TB

使用例 SAVE
SAVE CARDS

機 能 ［引数なし］カレント・プログラムをディスクに保存します。
［引数一つ］カレント・プログラムを，指定した名前でディスクに保存します。

SCREEN関数 QB

書 式 SCREEN(*row*, *column* [,*colorflag*])

機 能 指定した画面上の位置にある文字の，ASCIIコードや色を読み取ります。

SCREEN N88

書 式 SCREEN 〔〈画面モード〉〕〔, 〈画面スイッチ〉〕〔, アクティブ・ページ〕〔, ディスプレイ・ページ〕

機 能 グラフィック画面に対して種々のモードを設定します。

SCREENステートメント QB

書 式 SCREEN [*mode*] [, [*paletteswitch*]] [, [*apage*]] [, [*vpage*]]

機 能 ディスプレイ画面の設定を行ないます。

SCRIPT TB

書 式 SCRIPT PROCI

機 能 ［引数一つ］スクリプト・ファイルを実行します。

SEARCH N88

書 式 SEARCH (〈配列変数名〉, 〈整数表記〉〔, 〈開始添字〉〕〔, 〈ステップ値〉〕)

機 能 配列変数の中から指定された値を捜し出し，その要素の順位を得ます。

SEC関数 TB

書 式 SEC 〈数値式〉

機 能 〈数値式〉が示す角度のセカント（正割）の値を返します。

SEEK関数 QB

書 式 SEEK(*filenumber*)

機 能 ファイル内の現在入出力を行なっている位置を返します。

SEEKステートメント QB

書 式 SEEK[#] *filenumber*, *position*

機 能 ファイル内の次に入出力を行なう位置を設定します。

SELECT CASEステートメント QB

書 式

SELECT CASE *testexpression*
CASE *expressionlist1*
　[*statementblock-1*]
CASE *expressionlist2*
　[*statementblock-1*]
[CASE ELSE
　[*statementblock-n*]]
END SELECT

機 能 式の値に応じて，ステートメント・ブロックの中の一つを実行します。

SELECT CASE構文 TB

書 式 SELECT CASE構文：：SELECT CASE 〈選択式〉
CASE 〈case条件式〉
...
CASE 〈case条件式〉
...
...
CASE ELSE
...
END SELECT
〈選択式〉：：〈数値式〉
〈文字列式〉
〈case条件式〉：：〈case条件〉 .... 〈case条件〉
〈case条件〉：：〈定数〉
〈定数〉 TO 〈定数〉
IS 〈比較演算子〉〈定数〉
〈定数〉：：〈数値定数〉
〈ダブル・クォーテーション・マークで囲まれた文字列〉

機 能 〈選択式〉の値によって，制御を多方向に分岐させます。

SET (DISKモード) N88

書 式 1) SET 〈ドライブ番号〉, |"P" / "R"|
2) SET 〈ファイル・ディスクリプタ〉, |"P" / "R"|
3) SET # 〈ファイル番号〉, |"P" / "R"|

機 能 ファイル属性のセット・リセットを行ないます。

SET BACKGROUND COLOR文 TB

書 式 SET BACKGROUND COLOR 〈整数式〉
SET BACK 〈整数式〉
SET BACKGROUND COLOR 〈文字列式〉
SET BACK 〈文字列式〉

機 能 バックグラウンド・カラーを〈整数式〉または〈文字列式〉で指定した色に設定します。

SET COLOR文 TB

書 式 SET COLOR 〈整数式〉
SET COLOR 〈文字列式〉

機 能 フォアグラウンド・カラーを〈整数式〉または〈文字列式〉で指定した色に設定します。

SET COLOR MIX文 TB

書 式 SET COLOR MIX 〈整数式〉〈数値式１〉, 〈数値式２〉, 〈数値式３〉

機 能 〈整数式〉で指定した番号の色の赤，緑，青の成分を設定します。

SET CURSOR文 TB

書 式 SET CURSOR 〈文字列式〉
SET CURSOR 〈整数式１〉〈整数式２〉

機 能 〈文字列式〉でカーソルの状態を次のように設定します。
〈文字列式〉の値がOFF　カーソルを消します。
その他　カーソルを表示します。
または，〈整数式１〉〈整数式２〉で指定した行と桁の位置にカーソルを設定します。

SET DIRECTORY文 TB

書 式 SET DIRECTORY 〈文字列式〉

機 能 カレント・ディレクトリを〈文字列式〉で指定したディレクトリに変更します。

SET LANGUAGE文 TB

書 式 SET LANGUAGE 〈文字列式〉
SET LANG 〈文字列式〉

機 能 プログラムの中で出力するメッセージのモードを，〈文字列式〉で指定したモードに変更します。

SET MARGIN文　TB

書式　SET MARGIN〈整数式〉
SET #〈整数式1〉: MARGIN〈整数式2〉

機能　ウィンドウやチャネルのマージンを，〈整数式〉で指定した値に変更します。

SET MODE文　TB

書式　SET MODE〈文字列式〉

機能　現在のスクリーン・モードを，〈文字列式〉で指定したモードに変更します。

SET NAME文　TB

書式　SET NAME〈文字列式〉

機能　カレント・プログラムの名前を，〈文字列式〉で指定した名前に変更します。

SETMEM関数　QB

書式　SETMEN(*numeric-expression*)

機能　farヒープで使うメモリ容量を変更します。farヒープとは，farオブジェクトと内部テーブルを格納する領域のことです。

SET POINTER文　TB

書式　SET #〈整数式〉: POINTER〈レコード移動〉
SET #〈整数式〉:〈I/Oリカバリ〉
SET #〈整数式〉: POINTER〈レコード移動〉,〈I/Oリカバリ〉

〈レコード移動〉: : BEGIN
END
NEXT
SAME

〈I/Oリカバリー〉: : IF MISSING THEN〈ジャンプ〉
IF THERE THEN〈ジャンプ〉

〈ジャンプ〉: : EXIT DO
EXIT FOR
〈行番号〉

機能　ファイルのポインタを〈レコード移動〉で指定した位置に移動します。

BEGIN　ファイルの最初
END　ファイルの終わり
NEXT　次のレコード
SAME　処理されたばかりのレコード

SET RECORD文　TB

書式　SET #〈整数式1〉: RECORD〈整数式2〉

機能　ファイル・ポインタを，指定したレコードまたはバイトに設定します。

SET RECSIZE文　TB

書式　SET #〈整数式1〉: RECSIZE〈整数式2〉

機能　ファイルのレコード長をバイト単位で設定します。

SET TEXT JUSTIFY文　TB

書式　SET TEXT JUSTIFY〈文字列式1〉,〈文字列式2〉

機能　PLOT TEXT文で表示する文字列の水平と垂直の位置を〈文字列式1〉〈文字列式2〉で設定します。
〈文字列式1〉では，水平の位置を指定します。
LEFT　左端
RIGHT　右端
CENTER　中央
〈文字列式2〉では，垂直の位置を指定します。
TOP　上端
BOTTOM　下端
BASE　基線
HALF　中央

SET WINDOW文　TB

書式　SET WINDOW〈数値式1〉,〈数値式2〉,〈数値式3〉,〈数値式4〉

機能　グラフィックのウィンドウ座標を設定します。

SET ZONEWIDTH文　TB

書式　SET ZONEWIDTH〈整数式〉
SET #〈整数式1〉: ZONEWIDTH〈整数式2〉

機能　ウィンドウやファイルの領域幅を設定します。

SGN関数　N88/TB/QB

書式　SGN〈数値式〉

機能　〈数値式〉の値の「符号」を返します。

SHARE文　TB

書式　SHARE〈共用項目〉....〈共用項目〉
〈共用項目〉: :〈単一数値変数〉
〈単一文字列変数〉
〈配列〉〈添字範囲〉
#〈整数〉

機能　モジュールのプロシージャ間で共用できる変数，配列，チャネルを指定します。

SHAREDステートメント　QB

書式　SHARED*variable* [AS *type*] [, *variable* [AS *type*]] ...

機能　SUB，FUNCTIONプロシージャがモジュール・レベルで宣言した変数にアクセスできるようにします。変数はパラメータとして引き渡されません。

SHELLステートメント　QB

書式　SHELL [*commandstring*]

機能　実行中のBASICのプログラムを抜けて，.COM，.EXE，.BATプログラムや，DOSのコマンドを実行します。実行を終了すると，SHELLステートメントの次の行に制御が戻ります。

SIN　N88/TB/QB

書式　SIN (〈数式〉)

機能　正弦（サイン）を得ます。

SINH関数　TB

書式　(〈数値式〉)

機能　〈数値式〉の双曲線サインの値を返します。

SIZE関数　TB

書式　SIZE (〈配列引数〉,〈整数式〉)
SIZE (〈配列引数〉)

機能　引数が二つあるときは，〈配列引数〉で指定された配列の中の，〈整数式〉番目の次元の要素の数を返します。引数が一つのときは，配列全体の要素の総数を返します。

SJIS関数　TB

書式　SJIS〈整数式〉

機能　〈整数式〉で表されるJISコードを，対応するシフトJISコードに変換して返します。

SLEEPステートメント　QB

書式　SLEEP [*seconds*]

機能　プログラムの実行を一次中断します。

SOUNDステートメント　QB

書式　SOUND *frequency*, *duration*

機能　スピーカから音を発生させます。

SOUND文　TB

書式　SOUND〈数値式1〉,〈数値式2〉

機能　〈数値式1〉で指定した周波数（ヘルツ）の音を，〈数値式2〉で指定した秒数だけ出力します。

SPACE$関数　QB

書式　SPACE$(*n*)

機能　n個のスペースから成る文字列を返します。

SPACE$　N88

書式　SPACE$〈数式〉

機能 任意の長さの空白文字列を得ます。

SPC関数 N88

書式 SPC〈数式〉
機能 任意の数だけの空白を出力します。

SPC関数 QB

書式 SPC(*n*)
機能 PRINT, LPRINTステートメントの中でn個のスペースを表示します。

SPLIT TB

書式 SPLIT
SPLIT 10
機能 [引数なし] ファンクション・キーの機能を表示(非表示)します。
[引数一つ] スプリット・バーの位置を変更します。

SQR関数 N88/TB

書式 SQR〈数式〉
機能 平方根を得ます。

SQR関数 QB

書式 SQR(n)
機能 指定した数値の平方根を返します。

STATICステートメント QB

書式 STATIC *variablelist*
機能 単純変数や単純配列を, DEF FN関数, FUNCTIONプロシージャ, SUBプロシージャに対してローカルな変数として宣言し, 次のプロシージャ呼び出しまで, その値を保存します。

STATUS DIAL関数 N88

書式 STATUS DIAL([[#]〈電話機番号〉,]〈短縮番号〉)
機能 電話機に記憶されている電話番号の機能情報を調べます。

STATUS DIAL$関数 N88

書式 STATUS DIAL$([[#]〈電話機番号〉,]〈短縮番号〉)
機能 電話機に記憶されている電話番号を調べます。

STATUS ERROR関数 N88

書式 STATUS ERROR [([#]〈電話機番号〉)]
機能 モデムからのデータ受信時における通信エラーの有無を調べます。

STATUS LINE関数 N88

書式 STATUS LINE [([#]〈電話機番号〉)]
機能 着信があったかどうかを調べます。

STATUS MODE関数 N88

書式 STATUS MODE [([#]〈電話機番号〉)]
機能 電話機の現在のモードを調べます。

STATUS PLAY関数 N88

書式 STATUS PLAY (〈チャンネル番号〉)
機能 サウンド・バッファの未演奏データのバイト数を得ます。

STOP N88

書式 STOP
機能 プログラムの実行を一時中断し,コマンド・モードにもどります。

STOPステートメント TB/QB

書式 STOP
機能 プログラムの実行を中止します。

STOP ON/OFF/STOP N88

書式 1) STOP ON
2) STOP OFF
3) STOP STOP
機能 [STOP] キーおよび [CTRL] + [C] による割り込みの許可, 禁止, 停止を制御します。

STR$関数 N88/TB/QB

書式 STR$(〈数式〉)
機能 数値を文字列に変換します。

STRIG ON,OFF,STOPステートメント QB

書式 STRING(*n*) ON
STRING(*n*) OFF
STRING(*n*) STOP
機能 ジョイスティックのイベント・トラッピングをオン/オフしたり, 中断します。

STRING$関数 QB

書式 1 STRING$(*m*, *numeric-expression*)
2 STRING$(*m*, *strigexpression*)
機能 すべての文字が同じASCIIコードをもつ文字列か, すべての文字が指定した文字列の先頭文字からなる文字列を返します。

STRING$関数 N88

書式 STRING$(〈式〉, |〈文字式〉/〈数式〉|)
機能 任意の文字を任意の数だけ連結した文字列を得ます。

SUBステートメント QB

書式 SUB*globalname* [(*parameterlist*)] [STATIC]
[EXIT SUB]
END SUB
機能 SUBプロシージャの始めと終りを示します。

SUB構文 TB

書式 〈SUB構文〉::〈sub文〉
...〈文〉
END SUB
〈sub文〉::SUB〈識別子〉
SUB〈識別子〉(〈サブルーチン・パラメータ・リスト〉)
〈サブルーチン・パラメータ・リスト〉::〈サブルーチン・パラメータ〉....
〈サブルーチン・パラメータ〉
〈サブルーチン・パラメータ〉::〈単一数値変数〉
〈単一文字列変数〉
〈配列パラメータ〉
#〈整数〉
機能 CALL文で呼び出すサブルーチンを定義します。〈サブルーチン・パラメータ・リスト〉は, CALL文の引数と一致していなければなりません。
構文を途中で抜けるときには, EXIT SUB文を使います。

SWAP N88/QB

書式 SWAP〈変数〉,〈変数〉
機能 二つの変数の値を入れ換えます。

SWITCH TB

書式 SWITCH
SWITCH KNIGHT
機能 [引数なし] スイッチの処理を中止(再開)します。
[引数一つ] カレント・プログラムから指定したプログラムにスイッチします。

SYSTEMステートメント QB

書式 SYSTEM
機能 オープンしているすべてのファイルをクローズし, オペレーティング・システムに制御を戻します。

TAB関数 N88

書式 TAB(〈数式〉)
機能 出力対象行の位置まで空白を空けます。

TAB関数　QB

書式　TAB(*column*)
機能　PRINT，LPRINTステートメントの中で，データの表示位置を移動します。

TAB関数　TB

書式　TAB（〈整数式〉）
　　　TAB（〈整数式1〉，〈整数式2〉）
機能　TABが使えるのは，PRINT文の中だけです。TAB(c)を使うと，カーソルをc桁まで移動できます。TAB(r,c)を使うと，カーソルを現在のウィンドウのr行C桁に移動できます。

TAN　N88/TB/QB

書式　TAN（〈数値式〉）
機能　正接（タンジェント）を得ます。

TANH関数　TB

書式　TANH（〈数値式〉）
機能　〈数値式〉の双曲線タンジェントの値を返します。

TERM　N88

書式　TERM "〔COM:〕〔パリティ・チェック〉〔〈データビット長〉〔〈ストップ・ビット長〉〔〈Xパラメータ〉〔〈Sパラメータ〉〔〈DELコード受信処理〉〔〈リターン・キー送信処理〉〔〈C_Rコード受信処理〉〔〈日本語シフトコード〉〕〕〕〕〕〕〕〕〕"〔，〔〈通信方式〉〕〔，〈変数領域の大きさ〉〕〕
機能　システムをターミナル・モードにします。

TIME関数　TB

書式　TIME
機能　午前0時からの秒数を返します。引数はありません。

TIMER関数　QB

書式　TIMER
機能　午前0時からそれまでに経過した秒数を返します。

TIMER ON,OFF,STOPステートメント　QB

書式　TIMER ON
　　　TIMER OFF
　　　TIMER STOP
機能　タイマイベントのトラッピングをオン／オフしたり，中断します。

TIME$関数　N88

書式　1）TIME$
　　　2）TIME$="hh：mm：ss"
機能　時刻を得ます。

TIME$関数　QB

書式　TIME$関数
機能　オペレーティング・システムから現在の時刻を返します。

TIME$関数　TB

書式　TIME$
機能　24時間制で計った時間を，文字列で返します。引数はありません。

TIME$ステートメント　QB

書式　TIME$=*stringexpression*
機能　時刻を設定します。

TIME$ ON/OFF/STOP　N88

書式　1）TIME$ ON
　　　2）TIME$ OFF
　　　3）TIME$ STOP
機能　リアルタイム・タイマによる割り込みの許可，禁止，停止を制御します。

TO　TB

使用例　TO 300
機能　［引数一つ］エディット・ウィンドウの指定した行にカーソルを移動します。

TRACE文　TB

書式　TRACE ON
　　　TRACE ON TO #〈整数式〉
　　　TRACE OFF
機能　デバッグが実行状態になっているプログラム・ユニットで，ひとつひとつの文の実行結果を表示します。

TRIM$関数　TB

書式　TRIM$（〈文字列式〉）
機能　〈文字列式〉で表される文字列の先頭または後方に空白がある場合に，どちらの空白も取り除いて返します。

TRN関数　TB

書式　TRN（〈数値配列〉）
機能　〈数値配列〉で表される配列を転置して返します。2次元の数値配列でなければなりません。TRNが使えるのは，MAT代入文内だけです。

TRON/TROFF　N88/QB

書式　1）TRON
　　　2）TROFF
機能　プログラムの実行状態を追跡します。

TRUNCATE関数　TB

書式　TRUNCATE（〈数値式〉，〈整数式〉）
機能　TRUNCATE(x,n)は，xを小数点以下n桁までで切り捨てた値を返します。nが負の場合は10の｜n｜乗の桁までで切り捨てた値を返します。

TRY　TB

書式　TRY青，赤
機能　［引数二つ］カレント・プログラムの文字列をひとつひとつ確認しながら置換します。

TYPEステートメント　QB

書式　TYPE *usertype*
　　　　elementname AS *typename*
　　　　elementname AS *typename*
　　　　.
　　　　.
　　　　.
　　　END TYPE
機能　一つ以上の要素を含むユーザー定義のデータ型を宣言します。

UBOUND関数　QB

書式　UBOUND(*array* [, *dimention*])
機能　配列の指定した次元で使うことができる，添字の上限（最大値）を返します。

UBOUND関数　TB

書式　UBOUND（〈配列引数〉，〈整数式〉）
　　　UBOUND（〈配列引数〉）
機能　引数が2個あるときは，〈配列引数〉で表された配列について，〈整数式〉番目の次元の添字の最大値（上限）を返します。2番目の引数がないときは，〈配列引数〉で表された配列の1番目の添字の最大値（上限）を返します。その配列は1次元の配列でなければなりません。

UCASE$関数　TB

書式　UCASE$（〈文字列式〉）
機能　〈文字列式〉で表される文字列の中にある英字の小文字をすべて大文字に変換して返します。

UCASE$関数　QB

書式　UCASE$(*stringexpression*)

機能　すべての文字を大文字に変換した文字列式を返します。

UEVENTステートメント　QB

書式　1 UEVENT ON
2 UEVENT OFF
3 UEVENT STOP

機能　ユーザー定義イベントのトラッピングをオン／オフしたり，中断します。

UNLOCKステートメント　QB

書式　UNLOCK [#] *filenumber* [, {*record* | [*start*] TO*end*}]

機能　ファイルの一部に設定したロックを解除します。

UNPACKB関数　TB

書式　UNPACKB (〈文字列式〉, 〈整数式1〉, 〈整数式2〉

機能　PACKBサブルーチンでパック（圧縮）した文字列を整数に復元（アンパック）した値を返します。

UNSAVE　TB

書式　UNSAVE Myfile

機能　[引数一つ] で保存されているファイルをディスクから削除します。

UNSAVE文　TB

書式　UNSAVE 〈文字列式〉

機能　〈文字列式〉で指定したファイルを削除します。

USING$関数　TB

書式　USING$ (〈文字列式〉, 〈式〉 .... 〈式〉)
〈式〉：：〈数値式〉
〈文字列式〉

機能　PRINT USING文で生成される文字列と同じものを返します。書式制御文字列である〈文字列式〉が先頭にあり，表示される〈数値式〉や〈文字列式〉を示す〈式〉が続きます。

USR　N88

書式　USR〔〈番号〉〕(〈引数〉)

機能　メモリ上に用意された機械語関数を呼び出します。

VAL関数　N88/TB/QB

書式　VAL (〈文字列〉)

機能　文字列表記の数値を実際の数値に変換します。

VARPTR関数　N88

書式　1) VARPTR (〈変数名〉〔, 〈機能〉〕)
2) VARPTR (# 〈ファイル番号〉〔, 〈機能〉〕)

機能　変数の値が格納されているメモリ番地，ファイルに割り当てられているファイル・コントロール・ブロックの開始番地を得ます。

VARPTR, VARSEG関数　QB

書式　VARPTR(*variablename*)
VARSEG(*variablename*)

機能　指定した変数のアドレスを返します。

VARPTR$関数　QB

書式　VARPTR$(*variablename*)

機能　DRAW, PLAYステートメントで使う変数のアドレスを文字列の形で返します。

VIEW　N88

書式　VIEW (Sx1, Sy1)－(Sx2, Sy2)〔, 〈領域色〉〕〔, 〈境界色〉〕

機能　ディスプレイ画面上での表示領域（ビューポート）を指定します。

VIEW関数　N88

書式　VIEW (〈機能〉)

機能　現在のビューポートの設定位置を得ます。

VIEWステートメント　QB

書式　VIEW [[SCREEN] (*x1*, *y1*)－(*x2*, *y2*) [, [*color*] [, *border*]]]

機能　グラフィックスの出力範囲を定義します。

VIEW KEYS　TB

書式　VIEW KEYS

機能　[引数なし] 定義されているキーと定義内容を表示します。

VIEW PRINTステートメント　QB

書式　VIEW PRINT [*topline*TO *bottomline*]

機能　画面上のテキスト・ビューポートの境界を設定します。

VOICE　N88

書式　VOICE 〈音色番号〉, 〈整数型配列名〉

機能　FM音源の音色バンクを再定義します。

VOICE COPY　N88

書式　VOICE COPY 〈音色番号〉, 〈整数型配列名〉

機能　FM音源の音色パラメータを配列にコピーします。

VOICE INIT　N88

書式　VOICE INIT

機能　FM音源の音色パラメータを初期化します。

VOICE LFO　N88

書式　VOICE LFO 〈チャンネル番号〉〔, 〈波形〉〕〔, 〈SYNCディレイ・タイム〉〕〔, 〈速さ〉〕〔, 〈ピッチ変調深さ（微）〉〕〔, 〈振幅変調深さ〉〕〔, 〈ピッチ変調深さ（粗）〉〕

機能　各チャンネルの出力にLFO効果を与えます。

VOICE REG　N88

書式　VOICE REG 〈レジスタ番号〉, 〈式〉

機能　シンセサイザLSIのレジスタに値を設定します。

WAIT　N88

書式　WAIT 〈ポート番号〉, 〈式1〉〔〈式2〉〕

機能　コンピュータの入力ポートをモニタする間，プログラムの実行を停止します。

WAITステートメント　QB

書式　WAIT*port number*, *and-expression* [, *xor-expression*]

機能　入力ポートを調べる間，プログラムの実行を一時中断します。

WHEN構文　TB

書式　〈WHEN構文〉：：WHEN EXCEPTION IN
〈保護部〉
USE
〈ハンドラ部〉
END WHEN
または
WHEN EXCEPTION USE 〈ハンドラ名〉
〈保護部〉
END WHEN
〈保護部〉：：...〈文〉
〈ハンドラ部〉：：...〈文〉
〈ハンドラ名〉：：〈識別子〉

機能　実行時エラーを処理します。

WHILE～WEND　N88

書式　WHILE 〈論理式〉
⌇
WEND

機能　WHILEからWENDまでの区間中にある一連の文を，指定条件が満足されている間，繰り返して実行します。

QB＝Quick BASIC　TB＝True BASIC

WHILE...WENDステートメント

書式 WHILE *condition*
[*statements*]
WEND

機能 指定した条件が真である間，ループ内のステートメントを繰り返し実行します。

WIDTH　N88

書式 1) WIDTH 〈桁数〉〔,〈行数〉〕
2) WIDTH 〈ファイル・ディスクリプタ〉, 〈サイズ〉
3) WIDTH # 〈ファイル番号〉, 〈サイズ〉

機能 各種出力機器やファイルに対して1行の長さなどを指定します。

WIDTHステートメント　TB

書式 1 WIDTH [*colums*] [, *lines*]
2 WIDTH {#*filenumber* | *device*} , *width*
3 WIDTH LPRINT *width*

機能 ファイルやデバイスに出力する行の幅を割り当てたり，画面に表示する桁数や行数を変更します。

WIDTH LPRINT　N88

書式 WIDTH LPRINT 〈文字数〉

機能 プリンタに出力する1行あたりの文字数を設定します。

WINDOW　N88

書式 WINDOW (Wx1, Wy1) − (Wx2, Wy2)

機能 ビューポートに表示するワールド座標系内の領域を指定します。

WINDOW　N88

書式 WINDOW (〈機能〉)

機能 現在のウィンドウの設定位置を得ます。

WINDOWステートメント　QB

書式 WINDOW [[SCREEN] (*x1*, *y1*) − (*x2*, *y2*)]

機能 現在のビューポートの論理座標を定義します。

WINDOW文　TB

書式 WINDOW # 〈整数式〉

機能 オープンしているウィンドウの中からカレント・ウィンドウを選択します。

WRITE文　TB

書式 WRITE # 〈整数式〉: 〈式〉 .... 〈式〉
WRITE # 〈整数式〉, 〈書き出しオプション〉 ... 〈書き出しオプション〉: 〈式〉 .... 〈式〉
　〈書き出しオプション〉: : 〈レコード移動〉
　　　　　　　　IF THERE THEN 〈ジャンプ〉
　〈レコード移動〉: : BEGIN
　　　　　　　END
　　　　　　　NEXT
　　　　　　　SAME
　〈ジャンプ〉: : EXIT DO
　　　　　　EXIT FOR
　　　　　　〈行番号〉
　〈式〉: : 〈数値式〉
　　　　〈文字列式〉

機能 〈式〉の値を計算し，〈整数式〉で指定したファイルに書き出します。

WRITE　N88

書式 WRITE 〈式〉〔 {, | ;} 〈式〉…〕

機能 画面にデータを出力します。

WRITEステートメント　QB

書式 WRITE [*expressionlist*]

機能 画面にデータを出力します。

WRITE#　N88

書式 WRITE# 〈ファイル番号〉, 〈式〉〔 {, | ;} 〈式〉…〕

機能 ファイルにデータを書き出します。

WRITE#ステートメント　QB

書式 WRITE#*filenumber*, *expressionlist*

機能 シーケンシャル・ファイルにデータを書き込みます。

ZER配列定数　TB

書式 ZER 〈添字変更式〉
ZER

機能 要素がすべて0である数値配列を返します。ZERが使えるのは，MAT代入文のなかだけです。〈添字変更式〉があるときは，その〈添字変更式〉で指定した次元の配列が生成されます。〈添字変更式〉がないときは，元の配列の次元のままです。